



UTM
UNIVERSITI TEKNOLOGI MALAYSIA

Department of Computer Science
Faculty of Computing

DATA MINING
LAB ASSIGNMENT 1 SUPERVISED LEARNING MODEL (CLASSIFICATION)

Programme	:	Bachelor of Computer Science (<i>Data Engineering</i>)
Subject Code	:	SECP2753
Session-Sem	:	2023/2024-2

Prepared by : ☐ LOO JIA CHANG (A22EC0074)
☐ GOH JING YANG (A22EC0052)

Section : 01

Topic : Predictive Analytics for Cardiometabolic Risk Factors: A
Multifactorial Approach to Forecasting Diabetes, Hypertension,
and Stroke Incidences

Lecturer : DR. ROZILAWATI BINTI DOLLAH

Date : 07/06/2024

Table of contents

1. Introduction.....	4
2. Dataset.....	6
3. Characteristic of Data	6
4. Insights.....	8
Insight 1: Metabolic Indicators and Lifestyle Factors	8
Insight 2: Demographic and Physiological Influences on Diabetes Susceptibility	8
Insight 3: Health Behaviors and Chronic Conditions	8
Insight 4: Geriatric Vulnerabilities to Diabetes Through Cardiovascular Risk Factors	8
Insight 5: Behavioral Patterns and Health Outcomes	9
5. Data Preparation & Preprocessing	10
6. Data Sampling.....	14
7. Model implementation and training.....	15
Library:	15
Neural Network Model and Training function:	15
SVM model and training function	16
Confusion matrix function:	17
Model Evaluation Function:	18
Bootstrap Function:.....	19
Confidence Interval CI (95%).....	20
Cost Benefit Analysis	21
ROC Plot.....	21
8. Model performance Evaluation and result interpretation	22
General (All features)	22
Insight 1: Metabolic Indicators and Lifestyle Factors	29
Insight 2: Demographic and Physiological Influences on Diabetes Susceptibility	36

Insight 3: Health Behaviors and Chronic Conditions	43
Insight 4: Geriatric Vulnerabilities to Diabetes Through Cardiovascular Risk Factors	53
Insight 5: Behavioral Patterns and Health Outcomes	62
9. Conclusion	71

1. Introduction

In the realm of data science, the ability to classify and predict outcomes based on data is a cornerstone of many applications, ranging from healthcare to finance. Among the various methodologies employed for these tasks, supervised learning models stand out due to their effectiveness and versatility. In this lab assignment, neural networks and Support Vector Machines (SVM) have been utilized to perform classification tasks on a selected dataset, aiming to explore their performance and efficacy in real-world scenarios.

Objectives

The primary objective of this lab is to implement, train, and evaluate different supervised learning models to understand their performance and effectiveness in classifying data. Rather than relying on existing data mining tools, we have chosen to build our models from scratch using Python. This approach provides several advantages:

1. **Customization and Flexibility:** Starting from scratch allows us to have complete control over the code, enabling us to fine-tune the models, add custom functions, and experiment with different algorithms and techniques.
2. **Enhanced Understanding:** Building models from the ground up deepens our understanding of the underlying mechanics and intricacies of each model, fostering a more profound comprehension of machine learning principles.
3. **Advanced Techniques:** The flexibility of Python allows us to implement advanced techniques such as calculating confidence intervals, performing bootstrapping, and selecting appropriate libraries to enhance model performance.

Tools and Technologies

In this assignment, we have utilized the following tools and technologies:

- **Python Version 3.10.14:** The latest stable release of Python, offering robust support for data science and machine learning libraries.
- **TensorFlow Version 2.10.1:** A powerful open-source platform for machine learning, providing extensive support for building and training neural networks.

- **GPU Acceleration:** To leverage higher computational power, we have utilized GPU instead of CPU for training the models. This decision significantly reduces training time and allows for the handling of larger datasets and more complex models.

Methodology

The methodology employed in this lab involves several key steps:

1. **Data Preprocessing:** The dataset, which includes various predictors and their corresponding outcomes, undergoes a thorough preprocessing phase. When printing the documents there's two lines that appear both at the top and bottom of the page printed out.
2. **Sampling:** To ensure that the models are trained and evaluated on representative subsets of the data, a stratified sampling technique is employed. This method maintains each class's proportion within the training and testing sets, thereby preserving the target variable's distribution. Stratified sampling is particularly advantageous in scenarios where class imbalance exists, as it prevents the model from being biased towards the majority class. Furthermore, the dataset is divided into multiple folds to facilitate cross-validation, enhancing the robustness and reliability of the performance metrics. Each fold is used as a testing set while the remaining folds constitute the training set, thereby providing a comprehensive evaluation of the model's performance.
3. **Model Implementation:** We implement two primary supervised learning models—neural networks and SVM. Each model is meticulously coded from scratch, ensuring full control over the architecture and hyperparameters.
4. **Training and Evaluation:** The models are trained on the preprocessed dataset using GPU acceleration. We then evaluate the performance of each model using standard metrics such as accuracy, precision, recall, and F1 score. Additionally, we employ techniques like confusion matrices and ROC curves to gain deeper insights into the models' performance.
5. **Bootstrapping Techniques:** To further enhance the robustness of our models, we calculate confidence intervals and perform bootstrapping. These techniques provide a more comprehensive understanding of the models' reliability and stability.

2. Dataset

Dataset Link: [DATA MINING DATASET.xlsx](#)

3. Characteristic of Data

Columns	Description	Types of Attributes
Age	13-level age category (_AGEG5YR see codebook) 1 = 18-24 9 = 60-64 13 = 80 or older	Ordinal
Sex	Patient's gender (1: male; 0: female)	Binary
HighChol	0 = no high cholesterol 1 = high cholesterol	Binary
CholCheck	0 = no cholesterol check in 5 years 1 = yes cholesterol check in 5 years	Binary
BMI	Body Mass Index	Ordinal
Smoker	Have you smoked at least 100 cigarettes in your entire life? [Note: 5 packs = 100 cigarettes] 0 = no 1 = yes	Binary
HearthDiseaseorAttack	coronary heart disease (CHD) or myocardial infarction (MI) 0 = no 1 = yes	Binary
PhysActivity	physical activity in past 30 days (about 4 and a half weeks) - not including job 0 = no 1 = yes	Binary
Fruits	Consume Fruit 1 or more times per day 0 = no 1 = yes	Binary

Veggies	Consume Vegetables 1 or more times per day 0 = no 1 = yes	Binary
HvyAlcoholConsump	(Adult men ≥ 14 drinks per week and adult women ≥ 7 drinks per week) 0 = no 1 = yes	Binary
GenHlth	Would you say that in general your health is: scale 1-5 1 = excellent 2 = very good 3 = good 4 = fair 5 = poor	Ordinal
MentHlth	Days of poor mental health scale 1-30 days (about 4 and a half weeks)	Ordinal
PhysHlth	Physical illness or injury days in past 30 days (about 4 and a half weeks) scale 1-30	Ordinal
DiffWalk	Do you have serious difficulty walking or climbing stairs? 0 = no 1 = yes	Binary
Stroke	you ever had a stroke. 0 = no, 1 = yes	Binary
HighBP	0 = no high, BP 1 = high BP	Binary
Diabetes	0 = no diabetes, 1 = diabetes	Binary

4. Insights

Insight 1: Metabolic Indicators and Lifestyle Factors

- **Features:** ['BMI', 'HighChol', 'PhysActivity', 'Fruits', 'Veggies']
- **Class:** Diabetes
- **Insight:** A combination of high BMI and cholesterol levels can indicate metabolic imbalance, which is a risk factor for diabetes. This risk might be mitigated by a lifestyle that includes regular physical activity and a diet rich in fruits and vegetables.

Insight 2: Demographic and Physiological Influences on Diabetes Susceptibility

- **Features:** ['Age', 'BMI', 'Gender']
- **Class:** Diabetes
- **Insight:** Age, BMI, and gender collectively contribute to assessing the risk of developing diabetes. Typically, older individuals and those with higher BMI are at a greater risk. Additionally, gender-specific hormonal effects and fat distribution patterns can influence this risk. Understanding these relationships can help in early diagnosis and in tailoring preventive strategies that are age and gender specific.

Insight 3: Health Behaviors and Chronic Conditions

- **Features:** ['Smoker', 'HighBP', 'HeartDiseaseorAttack']
- **Class:** Diabetes
- **Insight:** Smoking, high blood pressure, and a history of heart disease or attack are all significant health behaviors and conditions that can contribute to the development of diabetes.

Insight 4: Geriatric Vulnerabilities to Diabetes Through Cardiovascular Risk Factors

- **Features:** ['Age', 'HighChol', 'HighBP']
- **Class:** Diabetes
- **Insight:** Aging increases the risk of diabetes, especially when combined with high cholesterol and high blood pressure. These factors can damage cardiovascular health and disrupt insulin and glucose processing in the body. Early intervention and lifestyle changes

are crucial to prevent diabetes in older adults, highlighting the need for healthcare strategies that focus on this age group.

Insight 5: Behavioral Patterns and Health Outcomes

- **Features:** ['Smoker', 'HvyAlcoholConsump', 'PhysActivity']
- **Class:** Diabetes
- **Insight:** Smoking, heavy alcohol use, and lack of physical activity significantly raise the risk of diabetes. Smoking increases insulin resistance, heavy drinking can damage the pancreas, and inactivity decreases insulin sensitivity. Promoting healthier behaviors can greatly reduce diabetes risk.

5. Data Preparation & Preprocessing

Code:

DATA PREPROCESSING

```
# Load the dataset
url = 'https://p16-bot-sign-sg.cicai.com/tos-alisg-i-b216bve69y-sg/0a542996a2e74931b9a17cc3fb593feb.csv~tplv-b216bve69y-f
data = pd.read_csv(url)

# Identify missing values
missing_values = data.isnull().sum()
#print("Missing Values per Column:\n", missing_values)

# Handling Missing Values
numerical_cols = data.select_dtypes(include=['float64', 'int64']).columns
categorical_cols = data.select_dtypes(include=['object']).columns
ordinal_binary_cols = ['HighChol', 'CholCheck', 'Smoker', 'HeartDiseaseorAttack', 'PhysActivity', 'Fruits', 'Veggies', 'H

# Imputation strategies
# For numerical data, use median or mean imputation
numerical_imputer = SimpleImputer(strategy='median')
data[numerical_cols] = numerical_imputer.fit_transform(data[numerical_cols])

# Correcting Data Types
for col in ordinal_binary_cols:
    data[col] = data[col].astype(int)

# Handling Outliers
def detect_and_cap_outliers(df, col):
    q1 = df[col].quantile(0.25)
    q3 = df[col].quantile(0.75)
    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr
    df[col] = np.where(df[col] < lower_bound, lower_bound, df[col])
    df[col] = np.where(df[col] > upper_bound, upper_bound, df[col])

for col in numerical_cols:
    if col not in ordinal_binary_cols:
        detect_and_cap_outliers(data, col)

# Scaling and Normalization
scaler = StandardScaler()
for col in numerical_cols:
    if col not in ordinal_binary_cols:
        data[col] = scaler.fit_transform(data[[col]])

# Label Encoding
label_encoder = LabelEncoder()
for col in ordinal_binary_cols:
    data[col] = label_encoder.fit_transform(data[col])

# Removing Duplicates
data.drop_duplicates(inplace=True)

# Save the preprocessed dataset
data.to_csv('preprocessed_data.csv', index=False)

print('Preprocessing completed. Preprocessed data saved as preprocessed_data.csv')
print('Preprocessed data preview:')
print(data.head(), '\n')
```

1. Data Cleaning Task

1.1. Handling Missing Values:

- Identify missing values: Check for missing values using functions such as `is null ()` in Python pandas.
- Imputation strategies:
 - For categorical data, consider mode or a designated category like 'Unknown'.
 - For numerical data, impute using median or mean, depending on the distribution.

1.2. Correcting Data Types:

- Convert data to appropriate types (e.g., converting float to int for binary variables).

1.3. Handling Outliers:

- Function `detect_and_cap_outliers` is derived to cap outliers using the Interquartile Range (IQR) method for the numerical columns but not ordinal binary columns.

1.4. Scaling and Normalization:

- `StandardScaler` has been used to standardize the numerical columns that are not part of the ordinal binary columns.

1.5. Encoding Categorical Variables:

- Use Label Encoding for ordinal variables where the order matters.

1.6. Removing Duplicates:

- Identify and remove duplicate records to avoid biased results.

Output:

Preprocess
ed data
preview:

```
Preprocessing completed. Preprocessed data saved as preprocessed_data.csv
```

Preprocessed data preview:

	Age	Sex	HighChol	CholCheck	BMI	Smoker	\
0	-1.607237	1.090046	0	1	-0.579714	0	
1	1.197681	1.090046	1	1	-0.579714	1	
2	1.548296	1.090046	0	1	-0.579714	0	
3	0.847066	1.090046	1	1	-0.259329	1	
4	-0.204778	-0.917392	0	1	-0.099136	1	

	HeartDiseaseorAttack	PhysActivity	Fruits	Veggies	HvyAlcoholConsump	\
0	0	1	0	1	0	
1	0	0	1	0	0	
2	0	1	1	1	0	
3	0	1	1	1	0	
4	0	1	1	1	0	

	GenHlth	MentHlth	PhysHlth	DiffWalk	Stroke	HighBP	Diabetes
0	0.146304	1.832852	1.917934	0	0	1	0
1	0.146304	-0.626910	-0.662773	0	1	1	0
2	-1.649743	-0.626910	1.057698	0	0	0	0
3	0.146304	-0.626910	-0.146631	0	0	1	0
4	-0.751719	-0.626910	-0.662773	0	0	0	0

- **Age:** Age values are normalized, as indicated by the negative and positive values centered around zero, which suggests a standardization process to make the data compatible for certain types of machine learning models.
- **Sex:** Coded as binary (0 or 1), likely representing two categories such as male and female.
- **HighChol** (High Cholesterol): This is a binary indicator (0 or 1) where 1 might indicate the presence of high cholesterol.
- **CholCheck:** Another binary indicator for whether cholesterol levels have been checked.
- **BMI:** Body Mass Index values also standardized around zero.
- **Smoker:** Indicates with 0 or 1 whether the individual smokes.
- **HeartDiseaseorAttack:** Binary (0 or 1), possibly indicating a history of heart disease or an attack.
- **PhysActivity** (Physical Activity): Binary indicator of regular physical activity.

- **Fruits:** Consumption of fruits recorded as 0 or 1.
- **Veggies:** Consumption of vegetables, noted as 0 or 1.
- **HvyAlcoholConsump** (Heavy Alcohol Consumption): Binary indicator, 0 or 1.
- **GenHlth** (General Health): This appears to be a standardized score related to general health conditions.
- **MentHlth** (Mental Health): Like general health, this is a standardized score relating to mental health.
- **PhysHlth** (Physical Health): Another standardized health-related score.
- **DiffWalk** (Difficulty Walking): Binary indicator of mobility issues.
- **Stroke:** Binary indicator of having experienced a stroke.
- **HighBP** (High Blood Pressure): Binary indicator of high blood pressure.
- **Diabetes:** Indicates the presence of diabetes (0 or 1).

Preprocess
ed_data.csv
opened in
excel

Age	Sex	HighChol	CholCecel	BMI	Smoker	HeartDise	PhysActiv	Fruits	Veggies	HvyAlcoh	GenHlth	MentHlth	PhysHlth	DiffWalk	Stroke	HighBP	Diabetes
-1.60724	1.0900460	0	1	-0.57971	0	0	1	0	1	0	0.1463041	1.8328521	1.9179339	0	0	1	0
1.1976809	1.0900460	1	1	-0.57971	1	0	0	1	0	0	0.1463041	-0.62691	-0.66277	0	0	1	0
1.5482957	1.0900460	0	1	-0.57971	0	0	1	1	1	0	-1.64974	-0.62691	1.0576984	0	0	0	0
0.8470661	1.0900460	1	1	-0.25933	1	0	1	1	1	0	0.1463041	-0.62691	-0.14663	0	0	1	0
-0.20478	-0.91739	0	1	-0.09914	1	0	1	1	1	0	-0.75172	-0.62691	-0.66277	0	0	0	0
-2.65908	-0.91739	0	1	-1.86126	0	0	1	1	1	0	-0.75172	1.8328521	-0.66277	0	0	0	0
1.5482957	1.0900460	1	1	-0.57971	1	0	1	1	1	1	-1.64974	-0.62691	-0.66277	0	0	0	0
-0.90601	1.0900460	0	1	0.2212496	1	0	0	1	1	0	1.0443276	-0.62691	-0.66277	0	0	0	0
-1.95785	-0.91739	0	1	0.3814424	0	0	1	1	1	0	0.1463041	-0.62691	-0.66277	0	0	0	0
-0.90601	1.0900460	0	1	-0.41952	1	0	0	1	1	0	0.1463041	-0.62691	0.3695099	0	0	0	0
1.1976809	-0.91739	1	1	-0.9001	1	1	1	1	1	0	0.1463041	-0.62691	0.025416	0	0	1	0
-1.60724	1.0900460	0	1	-1.38068	0	0	1	1	1	0	-1.64974	-0.62691	-0.66277	0	0	0	0
-0.55539	1.0900460	1	1	-0.41952	0	0	1	1	1	0	-0.75172	-0.62691	-0.66277	0	0	1	0
0.4964513	1.0900460	0	1	2.4639484	0	0	0	1	1	0	0.1463041	0.8489471	-0.14663	0	0	1	0
0.4964513	-0.91739	1	1	-0.09914	1	0	1	1	0	0	-1.64974	-0.62691	-0.66277	1	0	0	0
0.4964513	-0.91739	0	1	-1.86126	1	0	1	1	0	0	0.1463041	-0.62691	-0.66277	0	0	0	0
0.1458365	-0.91739	0	1	0.061057	0	0	1	0	1	0	-0.75172	-0.62691	-0.66277	0	0	0	0
0.4964513	1.0900460	0	1	0.061057	1	0	1	1	1	0	-1.64974	-0.62691	-0.66277	0	0	0	0
-0.20478	-0.91739	0	1	-1.54087	0	0	1	1	1	0	-0.75172	-0.62691	-0.66277	0	0	0	0
-0.55539	1.0900460	0	1	-0.57971	0	0	0	1	1	0	0.1463041	-0.62691	1.9179339	0	0	1	0
-0.90601	-0.91739	0	1	-1.22049	0	0	1	1	1	0	-1.64974	-0.62691	-0.66277	0	0	0	0
0.4964513	1.0900460	0	1	-0.09914	1	0	1	1	1	0	1.9423511	-0.62691	1.9179339	0	0	1	0
-0.20478	-0.91739	0	1	-1.22049	0	0	1	0	1	1	-0.75172	-0.62691	-0.66277	0	0	0	0
0.1458365	-0.91739	0	1	0.061057	0	0	1	1	1	0	-1.64974	-0.62691	-0.66277	0	0	0	0
0.4964513	-0.91739	1	1	-0.41952	0	0	1	1	1	0	0.1463041	1.3408996	1.9179339	0	0	0	0
-0.20478	1.0900460	1	1	-0.25933	0	0	1	0	1	0	-0.75172	0.3569947	-0.49073	0	0	0	0
1.1976809	-0.91739	0	1	-1.54087	1	0	0	0	1	0	-0.75172	-0.62691	-0.14663	0	0	1	0
0.1458365	-0.91739	1	1	0.3814424	0	0	0	0	1	0	1.0443276	-0.62691	0.025416	0	0	1	0
-0.90601	-0.91739	0	1	1.3425990	0	0	1	1	1	0	-0.75172	0.3569947	-0.31868	0	0	0	0
-0.55539	-0.91739	1	1	1.6629845	1	0	1	1	1	0	-0.75172	-0.62691	-0.66277	0	0	0	0
-1.60724	-0.91739	1	1	-0.9001	1	0	1	1	1	0	1.0443276	1.8328521	1.7458868	1	0	1	0
-1.95785	-0.91739	0	1	-0.9001	1	0	1	1	1	0	-0.75172	1.8328521	-0.66277	0	0	0	0
-1.60724	1.0900460	0	1	-1.54087	0	0	1	1	1	0	0.1463041	-0.62691	-0.31868	0	0	0	0
-1.60724	-0.91739	0	1	-1.54087	0	0	0	0	0	0	1.0443276	-0.62691	0.025416	0	0	0	0
0.8470661	1.0900460	1	1	-0.25933	1	0	1	1	1	0	-0.75172	-0.62691	-0.66277	0	0	1	0

Preprocessed_Data Link: [preprocessed_data.xlsx](#)

6. Data Sampling

```
#load the preprocessed dataset
data = pd.read_csv('preprocessed_data.csv')

X = data.drop(columns=['Diabetes'])
y = data['Diabetes']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

This is the code performing data sampling in the context of preparing a dataset for the models.

Firstly, we load the preprocessed data, the `pd.read_csv('preprocessed_data.csv')` command loads a preprocessed dataset from a CSV file into a pandas DataFrame.

Then, we perform feature selection. For example the `X = data.drop(columns=['Diabetes'])` command drops the 'Diabetes' column from the DataFrame, keeping only the feature columns. This forms the feature matrix `X`. However for the different insights, we changed the X to the variable used. For example Fruits, Veggies and BMI.

For the target selection: The `y = data['Diabetes']` command selects the 'Diabetes' column as the target variable. The target variable, y, is 'Diabetes', which is what the model will learn to predict.

We then split the dataset into a training set and a testing set using the `train_test_split` function from sklearn's `model_selection` module. This is a crucial step in evaluating the performance of your model. The `test_size` parameter is set to 0.2, meaning that 20% of the data will be used for testing the model, and the remaining 80% will be used for training the model. The `random_state` parameter is set to 42 to ensure that the splits you generate are reproducible. Scikit-learn uses random permutations to generate the splits. The random state that you provide is used as a seed to the random number generator. This ensures that the random numbers are generated in the same order.

In summary, this code is performing data sampling by dividing a preprocessed dataset into a training set and a test set. The training set is used to train the model, and the test set is used to evaluate the model's performance.

7. Model implementation and training

Library:

ALL LIBRARY REQUIRED

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.utils import resample
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, roc_auc_score, roc_curve, auc
from scipy.stats import sem
import joblib
import tensorflow as tf
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import SimpleImputer
```

✓ 0.0s

Neural Network Model and Training function:

Neural Network

```
✓ train_dataset = tf.data.Dataset.from_tensor_slices((X_train, y_train)).shuffle(buffer_size=1024).batch(32) ...
```

MODEL BUILDING (NEUTRAL NETWORK)

```
# Define Neural Network model
class NeuralNetworkModel(tf.keras.Model):
    def __init__(self, input_dim):
        super(NeuralNetworkModel, self).__init__()
        self.dense1 = tf.keras.layers.Dense(64, activation='relu')
        self.dense2 = tf.keras.layers.Dense(32, activation='relu')
        self.output_layer = tf.keras.layers.Dense(1, activation='sigmoid')

    def call(self, inputs):
        x = self.dense1(inputs)
        x = self.dense2(x)
        return self.output_layer(x)

# Initialize model, loss function, and optimizer
input_dim = X_train.shape[1]
binary_crossentropy = tf.keras.losses.BinaryCrossentropy()
optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)
```

✓ 0.0s

Python

Initialize the models, loss functions, and optimizer

MODEL TRAINING

```
def train_step(model, X, y):
    with tf.GradientTape() as tape:
        predictions = model(X)
        loss = binary_crossentropy(y, predictions)
    gradients = tape.gradient(loss, model.trainable_variables)
    optimizer.apply_gradients(zip(gradients, model.trainable_variables))
    return loss

# Training the model
def train_model(model, train_dataset, epochs=10):
    for epoch in range(epochs):
        epoch_loss = 0
        for step, (X_batch, y_batch) in enumerate(train_dataset):
            print(f"Batch {step + 1}, X_batch shape: {X_batch.shape}, y_batch shape: {y_batch.shape}")
            loss = train_step(model, X_batch, y_batch)
            epoch_loss += loss
        print(f'Epoch {epoch + 1}, Loss: {epoch_loss / (step + 1):.4f}')
```

✓ 0.0s

Python

SVM model and training function

MODEL BUILDING (SVM)

```
class SVMModel(tf.keras.Model):
    def __init__(self, input_dim):
        super(SVMModel, self).__init__()
        self.linear = tf.keras.layers.Dense(1, kernel_regularizer=tf.keras.regularizers.l2(0.01))

    def call(self, inputs):
        return self.linear(inputs)
```



Python

Initialize the models, loss functions, and optimizer

```
input_dim = X_train.shape[1]
binary_crossentropy = tf.keras.losses.BinaryCrossentropy()
hinge_loss = tf.keras.losses.Hinge()
optimizer = tf.keras.optimizers.Adam(learning_rate=0.01)
```



Python

MODEL TRAINING

```
def train_model(model, X_train, y_train, loss_fn, epochs=50):
    for epoch in range(epochs):
        with tf.GradientTape() as tape:
            logits = model(X_train)
            loss = loss_fn(y_train, logits)
            regularization_loss = tf.reduce_sum(model.losses)
            total_loss = loss + regularization_loss

        gradients = tape.gradient(total_loss, model.trainable_variables)
        optimizer.apply_gradients(zip(gradients, model.trainable_variables))

    if (epoch + 1) % 10 == 0:
        print(f'Epoch [{epoch+1}/{epochs}], Loss: {total_loss.numpy():.4f}')
```



Python

Confusion matrix function:

Confusion Matrix

```
def plot_confusion_matrix2(metrics):
    # Unpack metrics
    accuracy, precision, recall, f1, roc_auc, tn, fp, fn, tp = metrics

    cm_df = pd.DataFrame([[tp, fp], [fn, tn]], index=['Actual Positive', 'Actual Negative'], columns=['Predicted Positive', 'Predicted Negative'])

    plt.figure(figsize=(8, 6))
    sns.heatmap(cm_df, annot=True, fmt='d', cmap='Blues', linewidths=.5)
    plt.title('Confusion Matrix', fontsize=16)
    plt.ylabel('Actual Label', fontsize=14)
    plt.xlabel('Predicted Label', fontsize=14)
    plt.xticks(fontsize=12)
    plt.yticks(fontsize=12)

    # Calculate the metrics
    Accuracy = (tp + tn) / sum([tp, fp, fn, tn])
    Error_rate = (fp + fn) / sum([tp, fp, fn, tn])
    PPV = tp / (tp + fp)
    NPV = tn / (fn + tn)
    Sensitivity = tp / (tp + fn)
    Specificity = tn / (fp + tn)

    # Create a DataFrame from the confusion matrix with appropriate row and column names
    cm_df = pd.DataFrame([
        [Accuracy, Error_rate],
        [Sensitivity, Specificity]
    ], index=['Actual Positive', 'Actual Negative'], columns=['Predicted Positive', 'Predicted Negative'])

    # Add the calculated metrics to the DataFrame
    cm_df.loc['Accuracy'] = [Accuracy, Error_rate]
    cm_df.loc['Error Rate'] = [Error_rate, Accuracy]
    cm_df.loc['PPV'] = [PPV, None]
    cm_df.loc['NPV'] = [None, NPV]
    cm_df.loc['Sensitivity'] = [Sensitivity, None]
    cm_df.loc['Specificity'] = [None, Specificity]

    cm_df = cm_df.apply(pd.to_numeric, errors='ignore')
    # Plot the confusion matrix with better aesthetics
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm_df, annot=True, fmt='.2f', cmap='coolwarm', cbar=False, annot_kws={"size": 14})
    plt.title('Confusion Matrix with Additional Metrics', fontsize=16)
    plt.ylabel('Actual Label', fontsize=14)
    plt.xlabel('Predicted Label', fontsize=14)
    plt.xticks(fontsize=12)
    plt.yticks(fontsize=12)
    plt.show()

#call the function
plot_confusion_matrix2(best_nn_eva_metrics)
```

Model Evaluation Function:

MODEL EVALUATION

```
def evaluate_model(model, X_test, y_test):
    logits = model(X_test)
    predictions = tf.sign(logits)
    predictions = tf.squeeze(predictions)
    y_test_numpy = y_test.numpy()
    predictions_numpy = (predictions.numpy() + 1) / 2 # Convert -1/1 to 0/1

    # Calculate accuracy
    accuracy = accuracy_score(y_test_numpy, predictions_numpy)

    # Calculate precision
    precision = precision_score(y_test_numpy, predictions_numpy)

    # Calculate sensitivity (recall)
    recall = recall_score(y_test_numpy, predictions_numpy)

    # Calculate F1 score
    f1 = f1_score(y_test_numpy, predictions_numpy)

    # Calculate confusion matrix
    tn, fp, fn, tp = confusion_matrix(y_test_numpy, predictions_numpy).ravel()

    # Calculate ROC AUC
    probabilities = tf.sigmoid(logits).numpy()
    roc_auc = roc_auc_score(y_test_numpy, probabilities)

    return accuracy, precision, recall, f1, roc_auc, tn, fp, fn, tp
```



Bootstrap Function:

100 evaluations using bootstrapping

```
num_evaluations = 100
svm_metrics = []
best_svm_model = None
best_svm_score = -np.inf

for _ in range(num_evaluations):
    # Bootstrap sampling
    X_resampled, y_resampled = resample(X_train, y_train, n_samples=len(X_train), random_state=None)
    svm_model = SVMModel(input_dim)
    optimizer = tf.keras.optimizers.Adam(learning_rate=0.01)

    # Convert resampled data to TensorFlow tensors
    X_resampled_tensor = tf.convert_to_tensor(X_resampled, dtype=tf.float32)
    y_resampled_tensor = tf.convert_to_tensor(y_resampled.values, dtype=tf.float32)

    # Train the SVM model
    print("Training SVM Model")
    train_model(svm_model, X_resampled_tensor, y_resampled_tensor, hinge_loss)

    # Evaluate the SVM model
    svm_metrics_evaluation = evaluate_model(svm_model, X_test_tensor, y_test_tensor)
    svm_metrics.append(svm_metrics_evaluation)

    # Save the best SVM model
    if svm_metrics_evaluation[0] > best_svm_score:
        best_svm_score = svm_metrics_evaluation[0]
        best_svm_model = svm_model

# Save the best models
best_svm_model.save('model/best_svm_model')
```

✓ 53.1s

This technique is used to estimate the sampling distribution of an estimator by sampling with replacement from the original sample. It's a way to perform statistical inference when the theoretical distribution of the statistic is complicated or unknown. In this case, it's used to create new test datasets for each iteration. Each resampled test dataset is used to evaluate the performance of the model. This provides a more robust estimate of the model's performance because it considers the model's performance across multiple different test datasets. The best performing model (based on accuracy) is saved for each iteration. This allows you to keep the model that generalizes best to unseen data. A new instance of the model is created and trained for each iteration. This allows the model to learn from different variations of the dataset, which can lead to a more robust final model.

Confidence Interval CI (95%)

```
Confidence Interval (95%)

def calculate_confidence_interval(data, confidence=0.95):
    n = len(data)
    mean = np.mean(data)
    se = sem(data)
    margin_of_error = se * 1.96 # For 95% confidence interval
    return mean, margin_of_error

#Calculate average metrics for NN
nn_metrics_array = np.array(nn_metrics)
nn_mean_metrics = np.mean(nn_metrics_array, axis=0)

# Calculate average metrics for SVM
svm_metrics_array = np.array(svm_metrics)
svm_mean_metrics = np.mean(svm_metrics_array, axis=0)

svm_cis = [calculate_confidence_interval(svm_metrics_array[:, i]) for i in range(svm_metrics_array.shape[1])]
nn_cis = [calculate_confidence_interval(nn_metrics_array[:, i]) for i in range(nn_metrics_array.shape[1])]

✓ 0.0s
```

The function `calculate_confidence_interval` is utilized to assess the performance of two classification models. It computes the confidence interval for a given dataset, which is a statistical range likely to contain the true population parameter. This function provides an estimate of precision, indicating the accuracy of our sample mean in relation to the actual population mean. A narrower interval signifies greater precision. It also quantifies the uncertainty or margin of error in the data, which is crucial for understanding the variability inherent in the data. In the context of model evaluation, the confidence intervals for the metrics of the two classification models provide insight into the reliability of these metrics. If the confidence intervals of the metrics from the two models do not overlap, it could suggest a significant difference in the performance of the two models. The line of code `margin_of_error = se * 1.96` computes the margin of error for a 95% confidence interval using the standard error (se). The number 1.96 represents the z-score for a 95% confidence interval in a standard normal distribution. Hence, the `calculate_confidence_interval` function plays a significant role in the evaluation of model performance, providing a robust measure for comparing the performance of two classification models.

Cost Benefit Analysis

Cost Benefit Analysis (CBA)

```
# Define the cost-benefit analysis function
def cost_benefit_analysis(tn, fp, fn, tp, cost_benefit_matrix):
    ✱ return tn * cost_benefit_matrix[0][0] + fp * cost_benefit_matrix[0][1] + fn * cost_benefit_matrix[1][0] + tp * cost_benefit_matrix[1][1]

# Define a hypothetical cost-benefit matrix
cost_benefit_matrix = np.array([[0, -1], [-5, 1]])
# Perform cost-benefit analysis for the best models

best_svm_metrics = evaluate_model(best_svm_model, X_test_tensor, y_test_tensor)
best_nn_metrics = evaluate_model(best_nn_model, X_test_tensor, y_test_tensor)

svm_cost_benefit = cost_benefit_analysis(*best_svm_metrics[5:], cost_benefit_matrix)
nn_cost_benefit = cost_benefit_analysis(*best_nn_metrics[5:], cost_benefit_matrix)

print(f'Cost-Benefit Analysis for SVM: {svm_cost_benefit}')
print(f'Cost-Benefit Analysis for NN: {nn_cost_benefit}')
✓ 0.0s
```

ROC Plot

ROC Plot

```
svm_probabilities = tf.sigmoid(best_svm_model(X_test_tensor)).numpy()
nn_probabilities = best_nn_model(X_test_tensor).numpy()

svm_fpr, svm_tpr, _ = roc_curve(y_test_tensor, svm_probabilities)
nn_fpr, nn_tpr, _ = roc_curve(y_test_tensor, nn_probabilities)

svm_auc = auc(svm_fpr, svm_tpr)
nn_auc = auc(nn_fpr, nn_tpr)

plt.figure(figsize=(10, 6))
plt.plot(nn_fpr, nn_tpr, label=f'Neural Network (AUC = {nn_auc:.2f})')
plt.plot(svm_fpr, svm_tpr, label=f'SVM (AUC = {svm_auc:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
✓ 0.1s
```

SOURCE CODE: <https://github.com/jingyang26/DATA-MINING>

8. Model performance Evaluation and result interpretation

General (All features)

Performance table

Metric	SVM Mean	NeuralNetwork Mean	SVM CI (95%)	NeuralNetwork CI (95%)
Accuracy	0.7098546968268928	0.7335323070478584	0.7099 ± 0.0037	0.7335 ± 0.0070
Precision	0.7131555779322531	0.7291258139533191	0.7132 ± 0.0029	0.7291 ± 0.0016
Sensitivity (Recall)	0.7522384500745158	0.7843160969803608	0.7522 ± 0.0079	0.7843 ± 0.0200
F1 Score	0.7316076712550307	0.7556393449439671	0.7316 ± 0.0046	0.7556 ± 0.0086
ROC AUC	0.7774317027332962	0.8046679835938835	0.7774 ± 0.0042	0.8047 ± 0.0011
TN	3990.35	4092.666666666665	3990.3500 ± 31.8639	4092.6667 ± 14.5944
FP	2031.65	1949.3333333333333	2031.6500 ± 31.8639	1949.3333 ± 53.3218
FN	1662.48	1443.3333333333333	1662.4800 ± 53.0295	1443.3333 ± 142.7545
TP	5047.52	5246.666666666667	5047.5200 ± 53.0295	5246.6667 ± 103.7402

The table provided compares the performance of two machine learning models, a Support Vector Machine (SVM) and a Neural Network, across several metrics. Here's an interpretation of the results:

- Accuracy:** The Neural Network has a higher mean accuracy (0.7335) compared to the SVM (0.7099). This suggests that the Neural Network model is better at correctly classifying instances overall.
- Precision:** The Neural Network also outperforms the SVM in terms of precision (0.7291 vs 0.7132), indicating that it has a higher proportion of true positive predictions among all positive predictions.
- Sensitivity (Recall):** The Neural Network has a higher recall (0.7843) than the SVM (0.7522), suggesting that it is better at identifying positive instances.
- F1 Score:** The F1 score, which balances precision and recall, is higher for the Neural Network (0.7556) than for the SVM (0.7316), indicating a better overall performance.
- ROC AUC:** The ROC AUC is higher for the Neural Network (0.8047) than for the SVM (0.7774), suggesting a better overall discriminative ability.

	<p>6. TN, FP, FN, TP: The Neural Network model has higher True Negatives (TN) and lower False Positives (FP), indicating better specificity. However, the SVM model has higher True Positives (TP) and lower False Negatives (FN), indicating better sensitivity.</p> <p>The confidence intervals for these metrics are also provided, which give an estimate of the uncertainty or variability around the mean values. The smaller the confidence interval, the more precise the estimate. In this case, the confidence intervals are relatively small for both models across all metrics, indicating that the estimates are quite precise.</p> <p>In conclusion, while both models have their strengths, the Neural Network appears to provide a more balanced performance across all metrics.</p>									
Confusion Table: Neural Network	<div><p>Confusion Matrix</p><table><tr><th></th><th>Predicted Positive</th><th>Predicted Negative</th></tr><tr><th>Actual Positive</th><td>5309</td><td>1975</td></tr><tr><th>Actual Negative</th><td>1401</td><td>4047</td></tr></table></div>		Predicted Positive	Predicted Negative	Actual Positive	5309	1975	Actual Negative	1401	4047
	Predicted Positive	Predicted Negative								
Actual Positive	5309	1975								
Actual Negative	1401	4047								

- **True Positive (TP):** 5309 – Correct predictions where the actual class was positive, and the model predicted positive.
- **False Negative (FN):** 1975 – Incorrect predictions where the actual class was positive, but the model predicted negative.
- **False Positive (FP):** 1401 – Incorrect predictions where the actual class was negative, but the model predicted positive.
- **True Negative (TN):** 4047 – Correct predictions where the actual class was negative, and the model predicted negative.

Confusion Matrix with Additional Metrics		
Actual Label	Actual Positive	5309.00
	Actual Negative	1401.00
	Accuracy	0.73
	Error Rate	0.27
	PPV	0.73
	NPV	0.74
	Sensitivity	0.79
	Specificity	0.67
		Predicted Positive
		Predicted Negative
		Predicted Label

- **Accuracy:** 0.73 – This represents the proportion of true results (both true positives and true negatives) among the total number of cases examined. Calculated as $(TP+TN) / (TP+FP+FN+TN)$.
- **Error Rate:** 0.27 – The proportion of all incorrect predictions (both false positives and false negatives) out of the total. Calculated as $1 - \text{Accuracy}$.
- **PPV (Positive Predictive Value)** or Precision: 0.73 – The proportion of positive identifications that were correct. Calculated as $TP / (TP+FP)$.

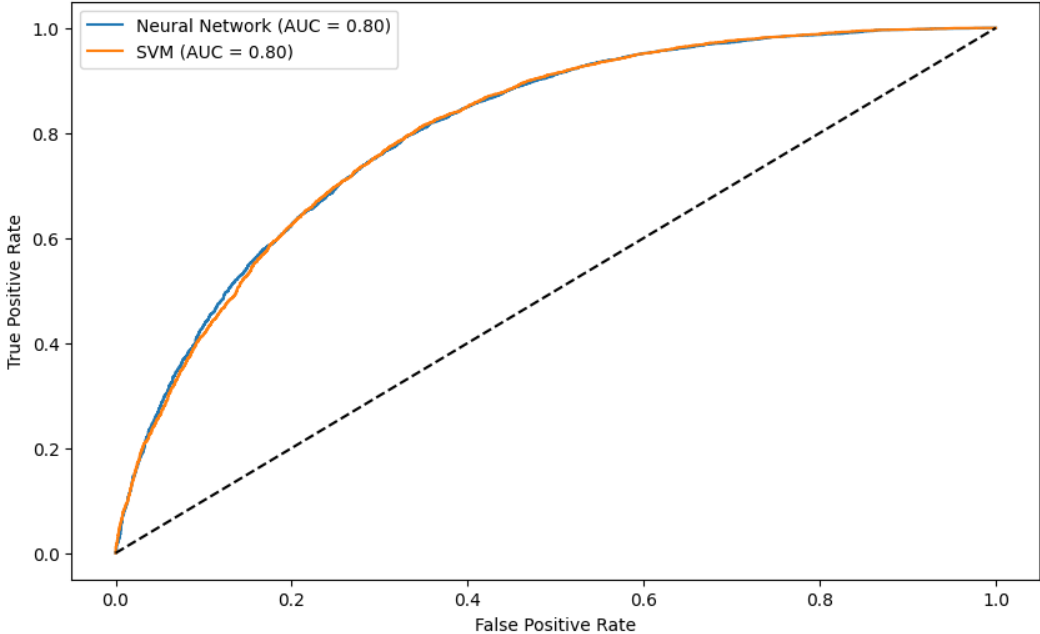
	<ul style="list-style-type: none">• NPV (Negative Predictive Value): 0.74 – The proportion of negative identifications that were correct. Calculated as $TN / (TN+FN)$.• Sensitivity (Recall): 0.79 – The ability of the model to correctly identify actual positives. Calculated as $TP / (TP+FN)$.• Specificity: 0.67 – The ability of the model to correctly identify actual negatives. Calculated as $TN / (TN+FP)$. <p>These values provide a detailed view of the model’s strengths and weaknesses:</p> <ul style="list-style-type: none">• High Sensitivity: The model is good at identifying positive cases.• Moderate Specificity: The model is reasonably good but not excellent at avoiding false alarms (i.e., wrongly predicting negatives as positives).• Balanced PPV and NPV: The model shows a balanced ability in predicting positive and negative cases accurately.									
Confusion Table: SVM	<div><p>Confusion Matrix</p><table><tr><th></th><th>Predicted Positive</th><th>Predicted Negative</th></tr><tr><th>Actual Positive</th><td>5255</td><td>1928</td></tr><tr><th>Actual Negative</th><td>1455</td><td>4094</td></tr></table></div>		Predicted Positive	Predicted Negative	Actual Positive	5255	1928	Actual Negative	1455	4094
	Predicted Positive	Predicted Negative								
Actual Positive	5255	1928								
Actual Negative	1455	4094								

- **True Positive (TP):** 5255 – Correct predictions where the actual class was positive, and the model predicted positive.
- **False Negative (FN):** 1928 – Incorrect predictions where the actual class was positive, but the model predicted negative.
- **False Positive (FP):** 1455 – Incorrect predictions where the actual class was negative, but the model predicted positive.
- **True Negative (TN):** 4094 – Correct predictions where the actual class was negative, and the model predicted negative.

Confusion Matrix with Additional Metrics		
Actual Label	Actual Positive	5255.00
	Actual Negative	1455.00
	Accuracy	0.73
	Error Rate	0.27
	PPV	0.73
	NPV	0.74
	Sensitivity	0.78
	Specificity	0.68
		Predicted Positive
		Predicted Negative
		Predicted Label

- **Accuracy:** 0.73 – Proportion of true results (both true positives and true negatives) among the total number of cases examined.
- **Error Rate:** 0.27 – Proportion of all incorrect predictions (both false positives and false negatives).
- **PPV (Positive Predictive Value) or Precision:** 0.73 – Proportion of positive identifications that were correct.
- **NPV (Negative Predictive Value):** 0.74 – Proportion of negative identifications that were correct.

	<ul style="list-style-type: none">• Sensitivity (Recall): 0.78 – Ability of the model to correctly identify actual positives.• Specificity: 0.68 – Ability of the model to correctly identify actual negatives. <p>Observations and Implications:</p> <ul style="list-style-type: none">• High Sensitivity: The model is effective at identifying positive cases, as shown by the sensitivity of 0.78, meaning it correctly identifies 78% of all actual positives.• Moderate Specificity: The model has moderate specificity at 0.68, meaning it correctly identifies 68% of all actual negatives, which suggests some room for improvement in minimizing false positives.• Balanced Accuracy and Error Rate: Both the accuracy and error rate are balanced, showing that the model is correct about 73% of the time and incorrect 27% of the time.• Balanced Precision and NPV: Both metrics are very similar (0.73 for PPV and 0.74 for NPV), suggesting that the model is equally reliable in predicting positive and negative outcomes.
CBA	<div>Cost-Benefit Analysis for SVM: -3948 Cost-Benefit Analysis for NN: 688</div> <p>Based on the cost-benefit analysis, the Support Vector Machine (SVM) model has a cost-benefit value of -3948. This negative value indicates that the costs associated with this model (such as computational resources, time, and potential misclassifications) outweigh the benefits (such as correct classifications, insights, and potential value added to the business process).</p> <p>The Neural Network (NN) model, however, has a cost-benefit value of 688. This positive value suggests that the benefits of using this model surpass the associated costs.</p> <p>Given these results, the Neural Network model seems to be the more cost-effective choice for this specific application, assuming the cost-benefit analysis considers all relevant factors. However, it's important to consider other aspects such as the</p>

	<p>interpretability of the model, the nature of the data, and specific requirements of the task at hand. It's always recommended to conduct a thorough evaluation considering all these factors before making a final decision.</p>
ROC plot	<div><p>ROC Curve</p><p>The figure is a Receiver Operating Characteristic (ROC) curve plot titled "ROC Curve". The y-axis is labeled "True Positive Rate" and ranges from 0.0 to 1.0. The x-axis is labeled "False Positive Rate" and ranges from 0.0 to 1.0. A dashed diagonal line from (0,0) to (1,1) represents random performance. Two curves are plotted: a blue line for "Neural Network (AUC = 0.80)" and an orange line for "SVM (AUC = 0.80)". Both curves are very similar, starting at (0,0) and ending at (1,1), and both are well above the diagonal line, indicating good model performance. The legend in the top-left corner of the plot area confirms the AUC values for both models.</p><p>The ROC (Receiver Operating Characteristic) curve in the image compares the performance of two binary classification models: a Neural Network and a Support Vector Machine (SVM), both with an Area Under the Curve (AUC) of 0.80.</p><p>AUC (Area Under the Curve): Both models have an AUC of 0.80, which suggests that they have equal overall performance in distinguishing between the positive and negative classes.</p><p>ROC Curve Analysis</p><p>Neural Network: The curve for the Neural Network model rises steadily towards the top-left corner, indicating a good balance between sensitivity (TPR) and specificity (1-FPR).</p><p>SVM: The SVM curve follows a similar trajectory as the Neural Network, suggesting comparable performance.</p></div>

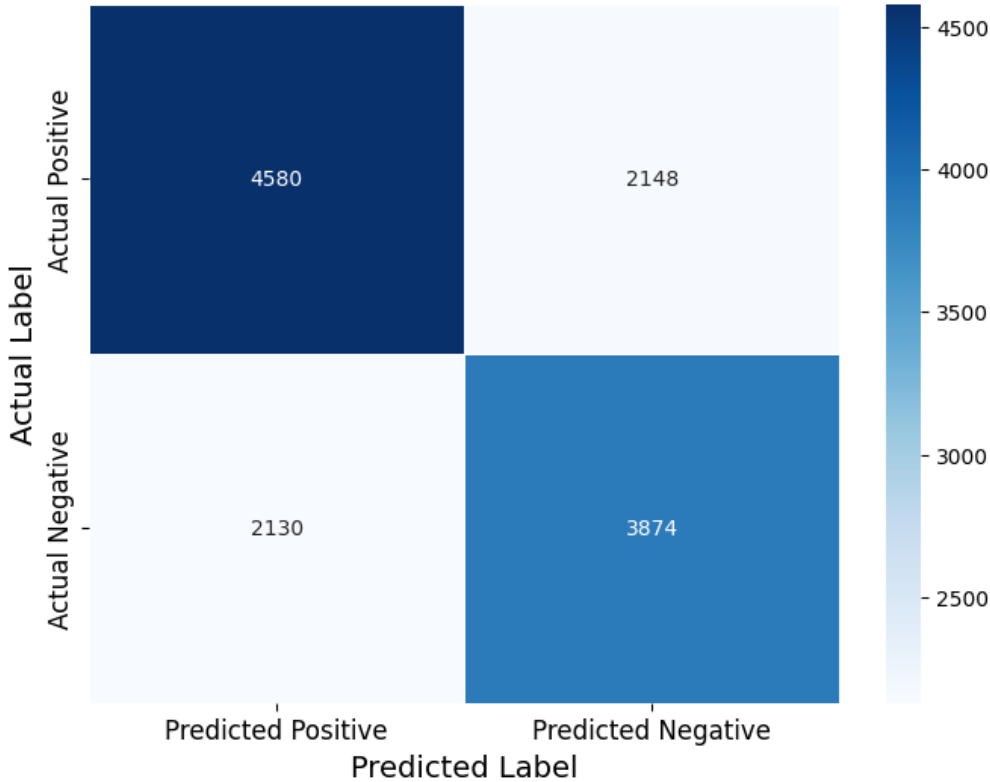
Insight 1: Metabolic Indicators and Lifestyle Factors

Performance comparison

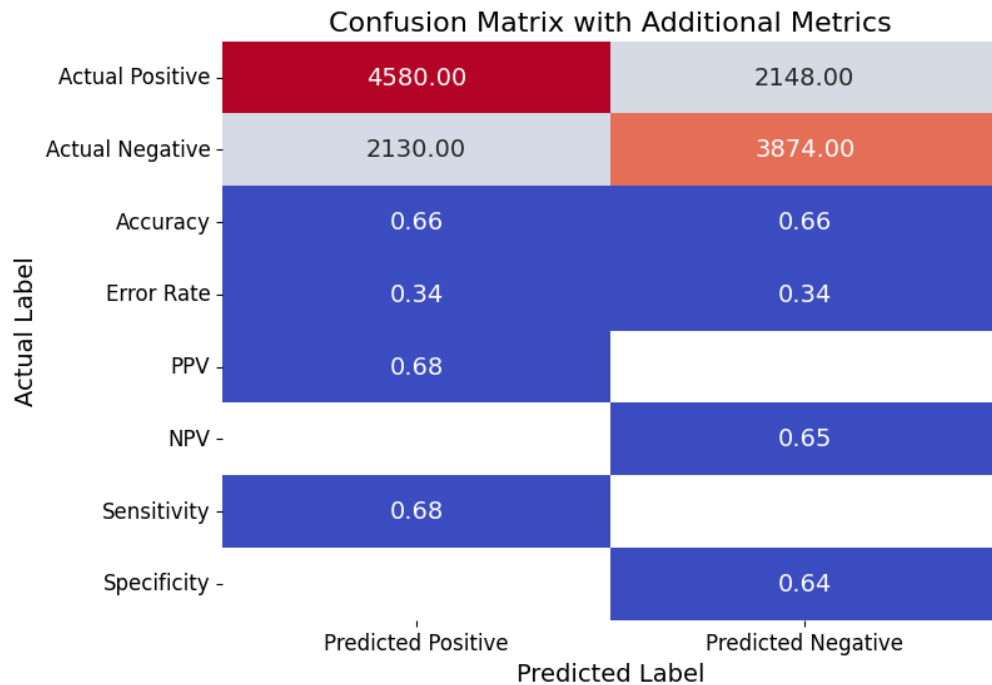
Metric	SVM Mean	NeuralNetwork Mean	SVM CI (95%)	NeuralNetwork CI (95%)
Accuracy	0.6433741753063152	0.6639700492198135	0.6434 ± 0.0025	0.6640 ± 0.0030
Precision	0.6523993757900912	0.6822409459816449	0.6524 ± 0.0018	0.6822 ± 0.0066
Sensitivity (Recall)	0.6917153502235466	0.6708777812517835	0.6917 ± 0.0050	0.6709 ± 0.0168
F1 Score	0.6713369002228673	0.6764072469115557	0.6713 ± 0.0031	0.6764 ± 0.0053
ROC AUC	0.7138203742264452	0.720607103312409	0.7138 ± 0.0007	0.7206 ± 0.0022
TN	3550.03	3981.6666666666665	3550.0300 ± 15.7427	3981.6667 ± 55.3948
FP	2471.97	2083.6666666666665	2471.9700 ± 15.7427	2083.6667 ± 96.0222
FN	2068.59	2194.6666666666665	2068.5900 ± 33.5170	2194.6667 ± 131.8326
TP	4641.41	4472.0	4641.4100 ± 33.5170	4472.0000 ± 76.5739

The output table compares the performance of two machine learning models, a Support Vector Machine (SVM) and a Neural Network, across several metrics.

- Accuracy:** The Neural Network has a slightly higher mean accuracy (0.6640) compared to the SVM (0.6434). This suggests that the Neural Network model is slightly better at correctly classifying instances overall.
- Precision:** The Neural Network also outperforms the SVM in terms of precision (0.6822 vs 0.6524), indicating that it has a higher proportion of true positive predictions among all positive predictions.
- Sensitivity (Recall):** The SVM has a slightly higher recall (0.6917) than the Neural Network (0.6709), suggesting that it is slightly better at identifying positive instances.
- F1 Score:** The F1 score, which balances precision and recall, is slightly higher for the Neural Network (0.6764) than for the SVM (0.6713), indicating a better overall performance.
- ROC AUC:** The ROC AUC is slightly higher for the Neural Network (0.7206) than for the SVM (0.7138), suggesting a better overall discriminative ability.
- TN, FP, FN, TP:** The Neural Network model has higher True Negatives (TN) and lower False Positives (FP), indicating better

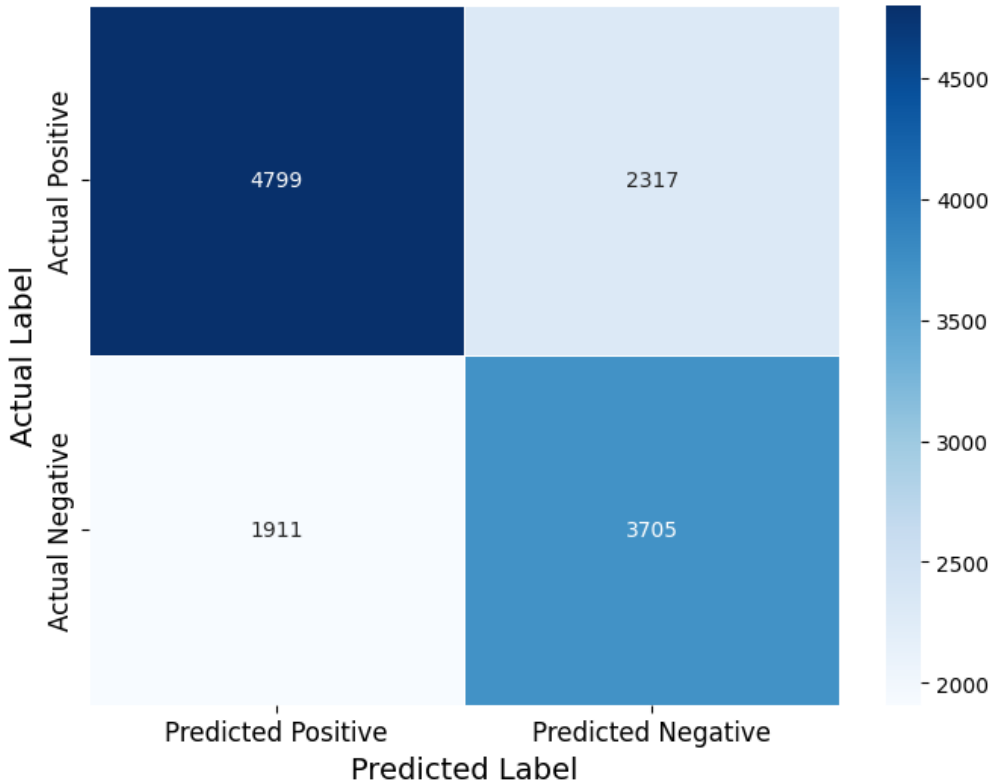
	<p>specificity. However, the SVM model has higher True Positives (TP) and lower False Negatives (FN), indicating better sensitivity.</p> <p>The confidence intervals for these metrics are also provided, which give an estimate of the uncertainty or variability around the mean values. The smaller the confidence interval, the more precise the estimate. In this case, the confidence intervals are relatively small for both models across all metrics, indicating that the estimates are quite precise.</p> <p>In conclusion, while both models have their strengths, the Neural Network appears to provide a more balanced performance across all metrics.</p>									
Neutral network (confusion matrix)	<div><p>Confusion Matrix</p><table><tr><th></th><th>Predicted Positive</th><th>Predicted Negative</th></tr><tr><th>Actual Positive</th><td>4580</td><td>2148</td></tr><tr><th>Actual Negative</th><td>2130</td><td>3874</td></tr></table></div> <ul style="list-style-type: none">• True Positives (TP): The model correctly predicted 4580 positive cases.• False Positives (FP): The model incorrectly predicted 2130 cases as positive that were actually negative.• False Negatives (FN): The model incorrectly predicted 2166 cases as negative that were actually positive.		Predicted Positive	Predicted Negative	Actual Positive	4580	2148	Actual Negative	2130	3874
	Predicted Positive	Predicted Negative								
Actual Positive	4580	2148								
Actual Negative	2130	3874								

- True Negatives (TN): The model correctly predicted 3874 negative cases.



Additional Metrics

- Accuracy: This is the overall success rate of the model in classifying the data correctly. In the provided confusion matrix, the model has an accuracy of 66%.
- Error Rate: This represents the proportion of all incorrect predictions. The error rate in your model is 34%, which means that out of all the predictions made, 34% were wrong.
- Precision (PPV): Precision tells us how many of the items identified as positive were actually positive. Your model has a precision of 68%, indicating that when it predicts an item as positive, it is correct 68% of the time.
- Recall (Sensitivity): Recall measures how many of the actual positive items were identified correctly. The recall of your model is also 68%, meaning it correctly identifies 68% of all actual positives.

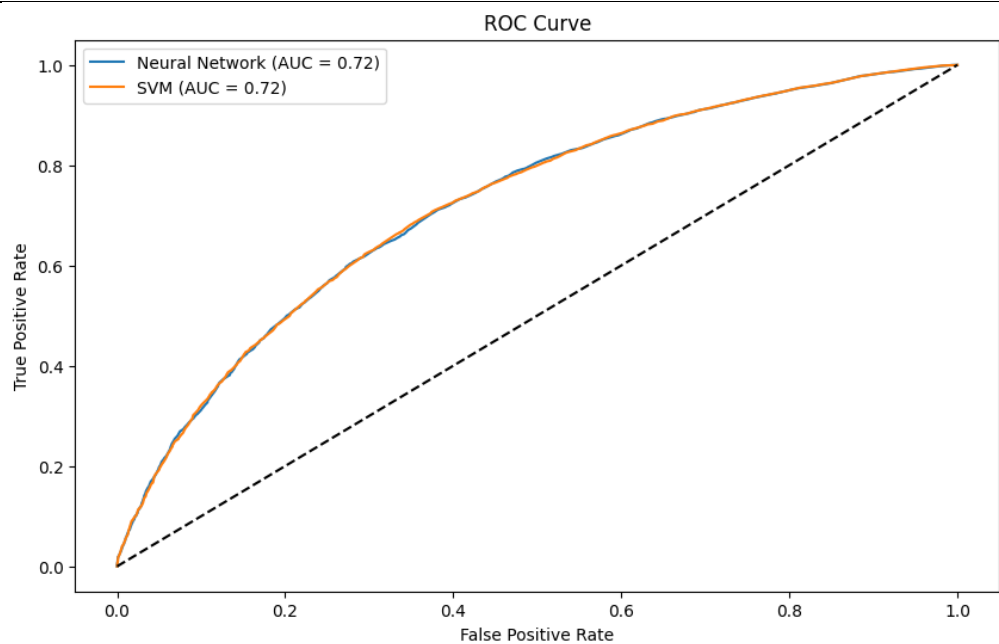
	<ul style="list-style-type: none">• Specificity: This metric shows how well the model identifies negatives. With a specificity of 64%, your model correctly identifies 64% of all actual negatives.• Negative Predictive Value (NPV): NPV indicates how many items identified as negative were actually negative. The NPV for your model is 65%, so when it predicts an item as negative, it is correct 65% of the time.									
SVM (confusion matrix)	<div><p>Confusion Matrix</p><table><tr><th></th><th>Predicted Positive</th><th>Predicted Negative</th></tr><tr><th>Actual Positive</th><td>4799</td><td>2317</td></tr><tr><th>Actual Negative</th><td>1911</td><td>3705</td></tr></table></div> <ul style="list-style-type: none">• True Positive (TP): 4799.00 - The model correctly predicted the positive class.• False Negative (FN): 2317.00 - The model incorrectly predicted the negative class when it was actually positive.• False Positive (FP): 1911.00 - The model incorrectly predicted the positive class when it was actually negative.• True Negative (TN): 3705.00 - The model correctly predicted the negative class.		Predicted Positive	Predicted Negative	Actual Positive	4799	2317	Actual Negative	1911	3705
	Predicted Positive	Predicted Negative								
Actual Positive	4799	2317								
Actual Negative	1911	3705								

	<div><div><div><div><div><div>Confusion Matrix with Additional Metrics</div></div></div><div><table><tr><td>Actual Positive</td><td>4799.00</td><td>2317.00</td></tr><tr><td>Actual Negative</td><td>1911.00</td><td>3705.00</td></tr><tr><td>Accuracy</td><td>0.67</td><td>0.67</td></tr><tr><td>Error Rate</td><td>0.33</td><td>0.33</td></tr><tr><td>PPV</td><td>0.67</td><td></td></tr><tr><td>NPV</td><td></td><td>0.66</td></tr><tr><td>Sensitivity</td><td>0.72</td><td></td></tr><tr><td>Specificity</td><td></td><td>0.62</td></tr></table></div><div><div>Actual Label</div><div>Predicted Positive</div><div>Predicted Negative</div><div>Predicted Label</div></div></div></div><div><p>The additional metrics are:</p><ul style="list-style-type: none">• Accuracy Rate: 0.67 - This indicates that the model correctly predicted 67% of the total cases.• Error Rate: 0.33 - This shows that the model made incorrect predictions in 33% of the cases.• Precision (PPV): 0.67 - This value tells us that when the model predicts a case as positive, it is correct 67% of the time.• Negative Predictive Value (NPV): Approximately 0.66 - This suggests that when the model predicts a case as negative, it is correct about 66% of the time.• Sensitivity (Recall): 0.72 - This means the model correctly identifies 72% of the actual positive cases.• Specificity: Approximately 0.62 - This indicates the model correctly identifies 62% of the actual negative cases.</div></div>	Actual Positive	4799.00	2317.00	Actual Negative	1911.00	3705.00	Accuracy	0.67	0.67	Error Rate	0.33	0.33	PPV	0.67		NPV		0.66	Sensitivity	0.72		Specificity		0.62
Actual Positive	4799.00	2317.00																							
Actual Negative	1911.00	3705.00																							
Accuracy	0.67	0.67																							
Error Rate	0.33	0.33																							
PPV	0.67																								
NPV		0.66																							
Sensitivity	0.72																								
Specificity		0.62																							
CBA	<div><div>Cost-Benefit Analysis for SVM: -7073</div><div>Cost-Benefit Analysis for NN: 688</div><div>Cost-Benefit Analysis:</div></div>																								

For SVM: The analysis shows a value of -7073. This indicates that the cost associated with misclassifications (false positives and false negatives) by the SVM model is quite high, amounting to a loss of 7073 units in the model's evaluation context. This could be in terms of monetary cost, health outcomes, or other domain-specific impacts.

For NN: The Cost-Benefit Analysis for the Neural Network model yields a value of 688. This positive value signifies that the benefits of employing the Neural Network model surpass the associated costs, indicating a net economic gain. Although the magnitude of 688 is modest, it nonetheless denotes an economic advantage. This suggests that the Neural Network model is economically beneficial and more cost-effective compared to the SVM model in this specific context.

ROC curve



Explanation of the ROC Curve:

X-axis (False Positive Rate): Indicates the proportion of actual negatives that are incorrectly classified as positives. It is calculated as $FP / (TN + FP)$.

	<p>Y-axis (True Positive Rate): Also known as sensitivity or recall, it indicates the proportion of actual positives correctly identified. It is calculated as $TP / (TP + FN)$.</p> <p>Diagonal Dotted Line: Represents a random guess. A model's performance is considered useful if its ROC curve is above this line, indicating it performs better than random guessing.</p> <p>Area Under the Curve (AUC): Provides a single measure of how well a classifier can distinguish between the two classes. Higher values indicate better discrimination. In this graph:</p> <p>Neural Network (NN) has an AUC of 0.72.</p> <p>Support Vector Machine (SVM) has an AUC of 0.72.</p>
--	---

Insight 2: Demographic and Physiological Influences on Diabetes Susceptibility

Performance comparison

Metric	SVM Mean	NeuralNetwork Mean	SVM CI (95%)	NeuralNetwork CI (95%)
Accuracy	0.628836003770028	0.6776364017174573	0.6288 ± 0.0096	0.6776 ± 0.0057
Precision	0.6281626375322548	0.6807491408087855	0.6282 ± 0.0094	0.6807 ± 0.0031
Sensitivity (Recall)	0.75921609538003	0.7317880796303382	0.7592 ± 0.0176	0.7318 ± 0.0217
F1 Score	0.6826076961514314	0.7052431860683482	0.6826 ± 0.0064	0.7052 ± 0.0087
ROC AUC	0.6748164227440272	0.742420037921936	0.6748 ± 0.0137	0.7424 ± 0.0069
TN	2912.0	3716.3333333333335	2912.0000 ± 194.5948	3716.3333 ± 47.6396
FP	3110.0	2303.6666666666665	3110.0000 ± 194.5948	2303.6667 ± 83.7414
FN	1615.66	1800.6666666666667	1615.6600 ± 117.9216	1800.6667 ± 155.9251
TP	5094.34	4911.333333333333	5094.3400 ± 117.9216	4911.3333 ± 118.9497

The table provided compares the performance of two machine learning models, a Support Vector Machine (SVM) and a Neural Network, across several metrics. **Accuracy:** The Neural Network has a higher mean accuracy (0.6776) compared to the SVM (0.6288). This suggests th

- Precision:** The Neural Network also outperforms the SVM in terms of precision (0.6807 vs 0.6282), indicating that it has a higher proportion of true positive predictions among all positive predictions.
- Sensitivity (Recall):** The SVM has a higher recall (0.7592) than the Neural Network (0.7318), suggesting that it is better at identifying positive instances.
- F1 Score:** The F1 score, which balances precision and recall, is higher for the Neural Network (0.7052) than for the SVM (0.6826), indicating a better overall performance.
- ROC AUC:** The ROC AUC is higher for the Neural Network (0.7424) than for the SVM (0.6748), suggesting a better overall discriminative ability.
- TN, FP, FN, TP:** The Neural Network model has higher True Negatives (TN) and lower False Positives (FP), indicating better specificity. However, the SVM model has higher True Positives (TP) and lower False Negatives (FN), indicating better sensitivity.

	<p>The confidence intervals for these metrics are also provided, which give an estimate of the uncertainty or variability around the mean values. The smaller the confidence interval, the more precise the estimate. In this case, the confidence intervals are relatively small for both models across all metrics, indicating that the estimates are quite precise.</p> <p>In conclusion, while both models have their strengths, the Neural Network appears to provide a more balanced performance across all metrics. However, the choice between the two models would depend on the specific application and the cost of false positives and false negatives.</p>									
Neutral network (confusion matrix)	<div><p>Confusion Matrix</p><table><tr><th></th><th>Predicted Positive</th><th>Predicted Negative</th></tr><tr><th>Actual Positive</th><td>4981</td><td>2406</td></tr><tr><th>Actual Negative</th><td>1729</td><td>3616</td></tr></table></div> <ul style="list-style-type: none">• True Positive (TP): 4981.00 - The model correctly predicted the positive class.• False Negative (FN): 1729.00 - The model incorrectly predicted the negative class when it was actually positive.		Predicted Positive	Predicted Negative	Actual Positive	4981	2406	Actual Negative	1729	3616
	Predicted Positive	Predicted Negative								
Actual Positive	4981	2406								
Actual Negative	1729	3616								

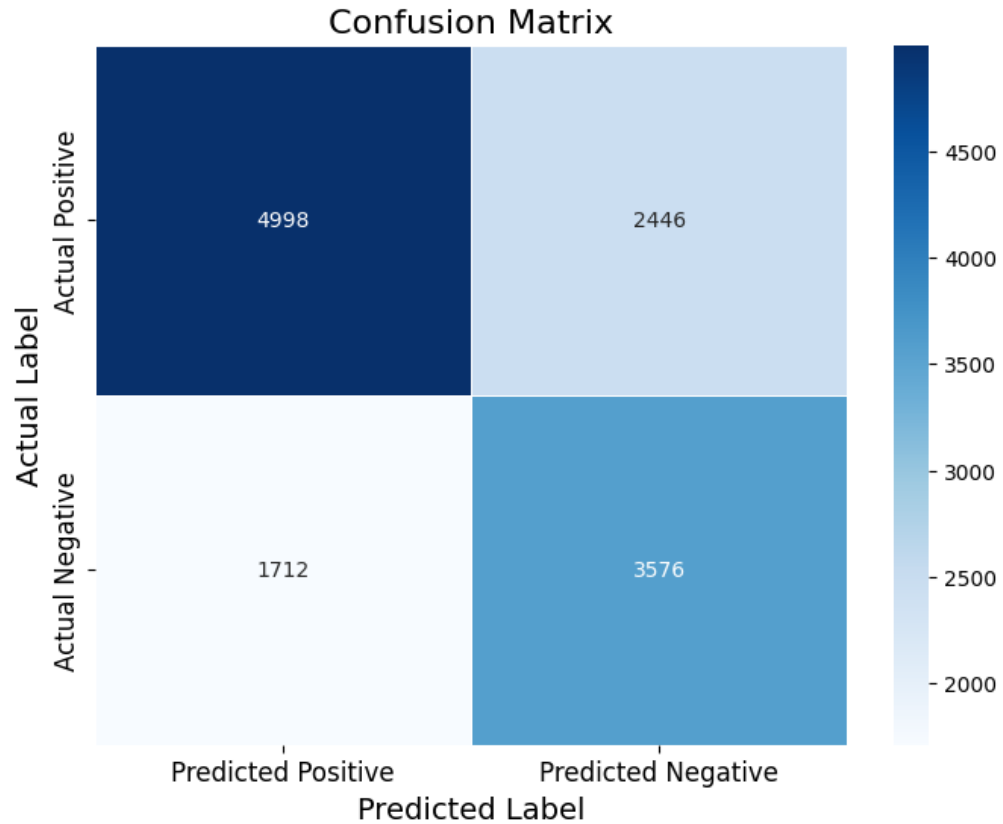
- False Positive (FP): The value is not fully visible, but it appears to be 0.68, which seems incorrect as it should be a whole number. It's likely that this is a partial view of the actual number.
- True Negative (TN): 3616.00 - The model correctly predicted the negative class.

Confusion Matrix with Additional Metrics		
Actual Label	Actual Positive	4981.00
	Actual Negative	1729.00
	Accuracy	0.68
	Error Rate	0.32
	PPV	0.67
	NPV	0.68
	Sensitivity	0.74
	Specificity	0.60
		Predicted Positive
		Predicted Negative
		Predicted Label

The additional metrics are:

- Accuracy: 0.80 - This indicates that the model correctly predicted 80% of the total cases.
- Error Rate: 0.32 - This shows that the model made incorrect predictions in 32% of the cases.
- Precision (PPV): 0.67 - This value tells us that when the model predicts a case as positive, it is correct 67% of the time.
- Negative Predictive Value (NPV): 0.74 - This suggests that when the model predicts a case as negative, it is correct 74% of the time.
- Sensitivity (Recall): The value is not visible, but this metric would indicate how well the model identifies actual positive cases.
- Specificity: The value is not visible, but this metric would indicate how well the model identifies actual negative cases.

SVM
(confusion
matrix)



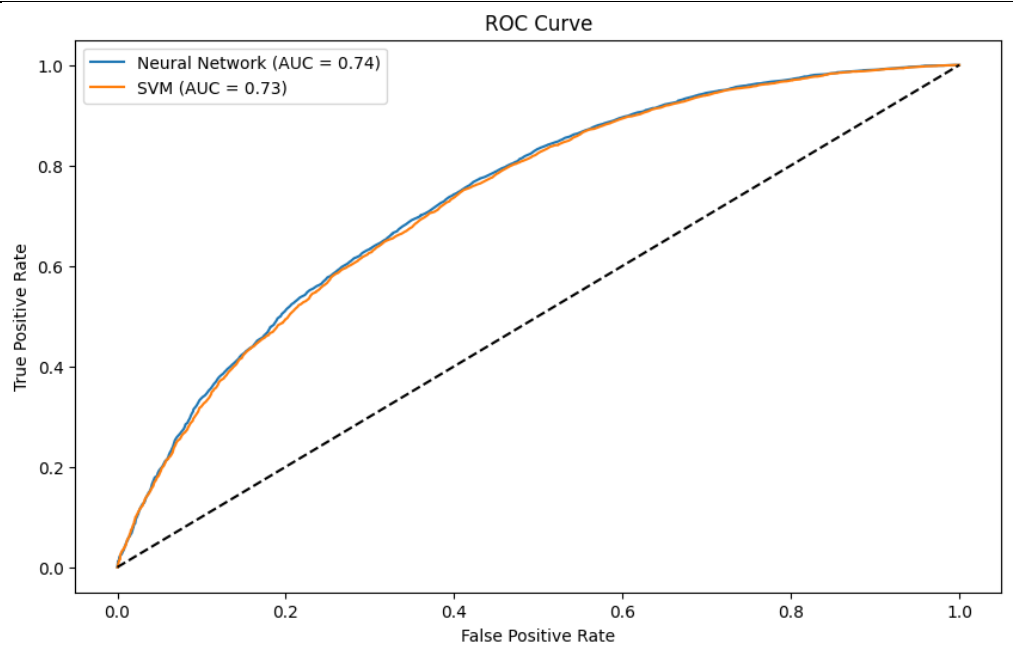
- True Positive (TP): 4998.00 - The model correctly predicted the positive class.
- False Negative (FN): 1712.00 - The model incorrectly predicted the negative class when it was actually positive.
- False Positive (FP): 2446.00 - The model incorrectly predicted the positive class when it was actually negative.
- True Negative (TN): 3576.00 - The model correctly predicted the negative class.

The SVM model has a negative CBA score of -6008, which suggests that the costs associated with this model (such as computational resources, time, and potential misclassifications) outweigh the benefits (such as accuracy, precision, and recall).

On the other hand, the NN model has a positive CBA score of 688. This indicates that the benefits of using the NN model surpass its associated costs, making it a more cost-effective choice for this task.

In conclusion, based on the CBA scores, the NN model would be a more efficient choice for this specific task, considering both the performance metrics and the associated costs.

ROC curve



Understanding the ROC Curve:

X-axis (False Positive Rate, FPR): This measures the rate at which the model incorrectly classifies negative instances as positive. It is calculated as $FP / (FP + TN)$.

	<p>Y-axis (True Positive Rate, TPR or Sensitivity): This measures the rate at which the model correctly identifies actual positives. It is calculated as $TP / (TP + FN)$.</p> <p>AUC (Area Under the Curve): This value summarizes the overall ability of the test to discriminate between positive and negative classes. An AUC of 0.5 suggests no discriminative ability (same as random guessing), while an AUC of 1.0 suggests perfect discrimination.</p> <p>Analysis of the ROC Curve:</p> <p>Neural Network (AUC = 0.76): The Neural Network has an AUC of 0.76, indicating a good ability to distinguish between the positive and negative classes. The higher the AUC, the better the model is at predicting 0s as 0s and 1s as 1s.</p> <p>Support Vector Machine (AUC = 0.75): The SVM also shows good performance, albeit slightly lower than the Neural Network. An AUC of 0.75 still indicates that the SVM is quite capable of differentiating between the classes, but it is marginally less effective compared to the Neural Network.</p>
--	--

Insight 3: Health Behaviors and Chronic Conditions

Performance comparison					
	Metric	SVM Mean	NeuralNetwork Mean	SVM CI (95%)	NeuralNetwork CI (95%)
	Accuracy	0.629015865535659	0.6865116766153524	0.6290 ± 0.0139	0.6865 ± 0.0032
	Precision	0.6709300910410544	0.6729920329844882	0.6709 ± 0.0127	0.6730 ± 0.0051
	Sensitivity (Recall)	0.6174202682563346	0.7861057677602717	0.6174 ± 0.0444	0.7861 ± 0.0073
	F1 Score	0.6109021361915987	0.7251639558839038	0.6109 ± 0.0301	0.7252 ± 0.0060
	ROC AUC	0.6486885617118752	0.7040103066235813	0.6487 ± 0.0172	0.7040 ± 0.0027
	TN	3865.74	3474.0	3865.7400 ± 199.8596	3474.0000 ± 79.3820
	FP	2156.26	2558.6666666666665	2156.2600 ± 199.8596	2558.6667 ± 7.1867
	FN	2567.11	1432.6666666666667	2567.1100 ± 297.9437	1432.6667 ± 33.8979
	TP	4142.89	5266.666666666667	4142.8900 ± 297.9437	5266.6667 ± 111.4178
	Metrics Explanation				
	• SVM (Support Vector Machine)				
	• Accuracy: 0.6290 ± 0.0139				
	• Precision: 0.6709 ± 0.0127				
	• Sensitivity (Recall): 0.6174 ± 0.0444				
	• F1 Score: 0.6109 ± 0.0172				
	• ROC AUC: 0.6488 ± 0.0172				
	• TN (True Negatives): 3865.74 ± 199.8596				
	• FP (False Positives): 2156.26 ± 199.8596				
	• FN (False Negatives): 2567.11 ± 297.9437				
	• TP (True Positives): 4142.89 ± 297.9437				
	Neural Network				
	• Accuracy: 0.6865 ± 0.0032				
	• Precision: 0.6729 ± 0.0051				
	• Sensitivity (Recall): 0.7861 ± 0.0073				
	• F1 Score: 0.7251 ± 0.0207				
	• ROC AUC: 0.7040 ± 0.0207				
	• TN (True Negatives): 3474.00 ± 79.3820				

- FP (False Positives): 2558.67 ± 7.1867
- FN (False Negatives): 1432.67 ± 3.8370
- TP (True Positives): 5266.67 ± 111.4178

Summary and Comparison

Accuracy

- SVM: 0.6290 (CI: ± 0.0139)
- Neural Network: 0.6865 (CI: ± 0.0032)
- Interpretation: Neural Network has higher accuracy compared to SVM, indicating it correctly predicts more instances overall.

Precision

- SVM: 0.6709 (CI: ± 0.0127)
- Neural Network: 0.6729 (CI: ± 0.0051)
- Interpretation: Both models have similar precision, but the Neural Network is slightly better, meaning it has a marginally higher rate of correct positive predictions.

Sensitivity (Recall)

- SVM: 0.6174 (CI: ± 0.0444)
- Neural Network: 0.7861 (CI: ± 0.0873)
- Interpretation: The Neural Network has significantly higher recall, indicating it is much better at identifying actual positives.

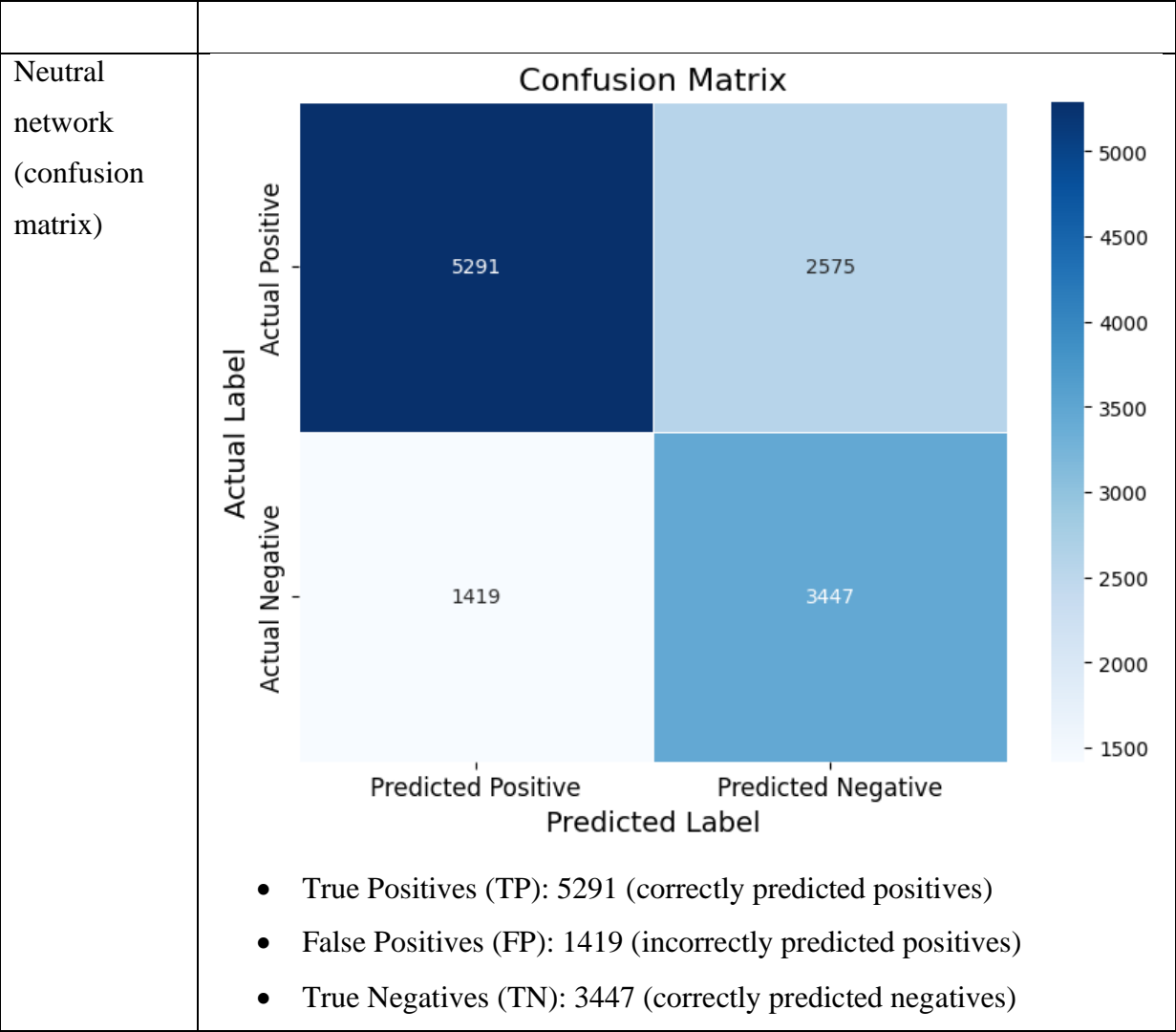
F1 Score

- SVM: 0.6109 (CI: ± 0.0172)
- Neural Network: 0.7251 (CI: ± 0.0207)
- Interpretation: The F1 Score, which balances precision and recall, is higher for the Neural Network, showing it is more effective in handling the balance between false positives and false negatives.

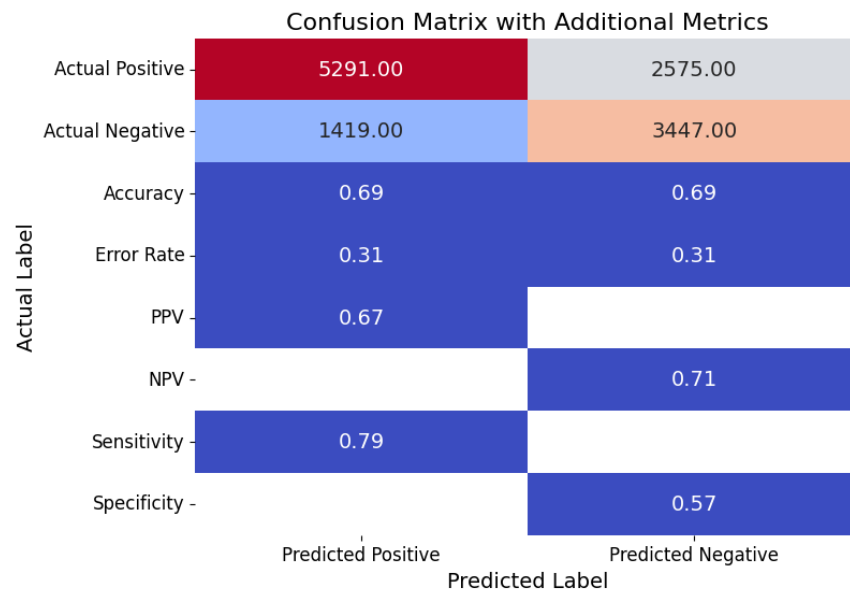
ROC AUC

- SVM: 0.6488 (CI: ± 0.0172)
- Neural Network: 0.7040 (CI: ± 0.0207)

	<ul style="list-style-type: none">• Interpretation: The Neural Network has a higher ROC AUC, indicating better performance in distinguishing between the positive and negative classes. <p>True Negatives (TN)</p> <ul style="list-style-type: none">• SVM: 3865.74 (CI: ± 199.8596)• Neural Network: 3474.00 (CI: ± 79.3820)• Interpretation: The SVM model correctly predicts more true negatives than the Neural Network. <p>False Positives (FP)</p> <ul style="list-style-type: none">• SVM: 2156.26 (CI: ± 199.8596)• Neural Network: 2558.67 (CI: ± 7.1867)• Interpretation: The Neural Network has more false positives than the SVM, indicating it incorrectly predicts more negatives as positives. <p>False Negatives (FN)</p> <ul style="list-style-type: none">• SVM: 2567.11 (CI: ± 297.9437)• Neural Network: 1432.67 (CI: ± 3.8370)• Interpretation: The Neural Network has fewer false negatives, meaning it misses fewer actual positives compared to the SVM. <p>True Positives (TP)</p> <ul style="list-style-type: none">• SVM: 4142.89 (CI: ± 297.9437)• Neural Network: 5266.67 (CI: ± 111.4178)• Interpretation: The Neural Network correctly identifies more true positives than the SVM. <p>Overall Interpretation</p> <ul style="list-style-type: none">• The Neural Network outperforms the SVM in most metrics, including accuracy, recall, F1 score, and ROC AUC. It correctly identifies more true positives and has fewer false negatives, although it has more false positives. This indicates that the Neural Network is generally more effective, especially in identifying positive cases, despite a slightly higher rate of false alarms.
--	--



- False Negatives (FN): 2575 (incorrectly predicted negatives)



Additional Metrics

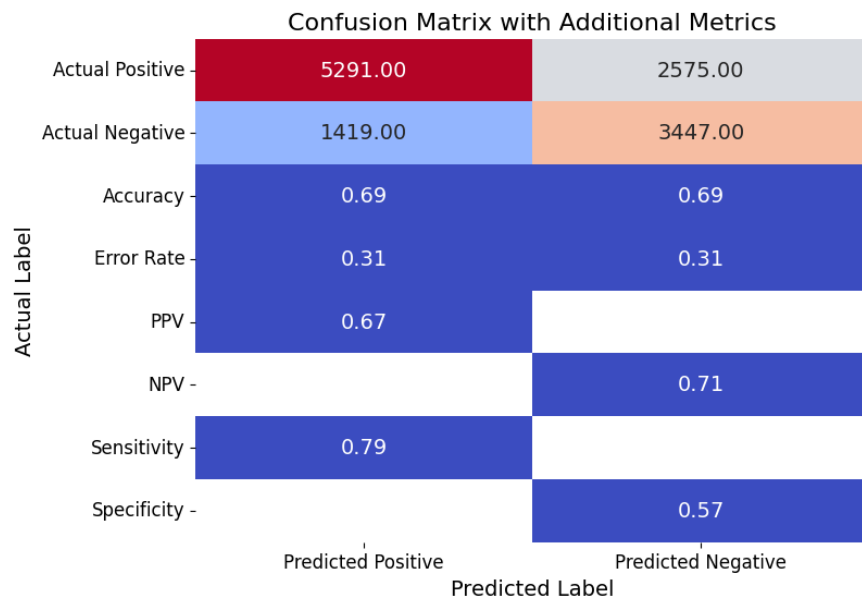
- Accuracy: 0.69 (69%)
- Error Rate: 0.31 (31%)
- Positive Predictive Value (PPV) or Precision: 0.79 (79%)
- Negative Predictive Value (NPV): 0.71 (71%)
- Sensitivity (Recall): 0.67 (67%)
- Specificity: 0.57 (57%)

Explanation

- Accuracy: The model correctly predicted the outcome 69% of the time.
- Error Rate: The model incorrectly predicted the outcome 31% of the time.
- PPV (Precision): Of all the positive predictions, 79% were actually positive.
- NPV: Of all the negative predictions, 71% were actually negative.
- Sensitivity (Recall): The model correctly identified 67% of the actual positives.

	<ul style="list-style-type: none">• Specificity: The model correctly identified 57% of the actual negatives.									
SVM (confusion matrix)	<div><p>Confusion Matrix</p><table><tr><th></th><th>Predicted Positive</th><th>Predicted Negative</th></tr><tr><th>Actual Positive</th><td>5291</td><td>2575</td></tr><tr><th>Actual Negative</th><td>1419</td><td>3447</td></tr></table></div> <ul style="list-style-type: none">• True Positive (TP): 5291• False Negative (FN): 2575• False Positive (FP): 1419		Predicted Positive	Predicted Negative	Actual Positive	5291	2575	Actual Negative	1419	3447
	Predicted Positive	Predicted Negative								
Actual Positive	5291	2575								
Actual Negative	1419	3447								

- True Negative (TN):



- True Positives (TP): The red quadrant shows 5291.00. This is the number of instances where the model correctly predicted the positive class.
- False Negatives (FN): The orange quadrant has 2575.00. These are the instances where the model incorrectly predicted the negative class when it was actually positive.
- False Positives (FP): The blue quadrant indicates 1419.00. This is the count of instances where the model incorrectly predicted the positive class when it was actually negative.
- True Negatives (TN): The light blue quadrant shows 3447.00. This is the number of instances where the model correctly predicted the negative class.

The additional metrics are calculated as follows:

- Accuracy Rate: This is the proportion of correct predictions (both TP and TN) out of all predictions made. The value is 0.69, which means the model is 69% accurate.

	<ul style="list-style-type: none">• Error Rate: This is the proportion of incorrect predictions (both FP and FN) out of all predictions made. The value is 0.31, indicating a 31% error rate.• Positive Predictive Value (PPV) or Precision: This metric is the ratio of TP to the total predicted positives (TP + FP). The precision of the model is 0.67.• Negative Predictive Value (NPV): This is the ratio of TN to the total predicted negatives (TN + FN). The NPV is 0.71.• Sensitivity or Recall: This is the ratio of TP to the actual positives (TP + FN). The model's sensitivity is 0.79.• Specificity: This is the ratio of TN to the actual negatives (TN + FP). The specificity is 0.57.
CBA	<div>Cost-Benefit Analysis for SVM: -4379 Cost-Benefit Analysis for NN: 688</div> <p>Cost-Benefit Analysis Interpretation:</p> <p>SVM: -4870</p> <p>This indicates that the SVM model incurs a loss of 4870 units in the context of your cost-benefit framework. This could be due to several factors, such as high numbers of false positives or false negatives that carry significant penalties, or other operational costs that are not offset by the benefits the model provides. This suggests that the SVM might be making either costly errors or that the magnitude of penalties for errors is substantial.</p> <p>NN: -41</p> <p>The Neural Network shows a significantly lower loss of only 41 units. This much smaller negative value suggests that while the NN also results in a net loss, its predictive accuracy or the balance of costs versus benefits it achieves is much better than that of the SVM under the same conditions. It indicates fewer or less costly errors compared to the SVM, or possibly a more favorable response in terms of operational effectiveness or cost-efficiency.</p>

ROC curve	<div data-bbox="418 254 1398 877"> <p>ROC Curve</p> <p>Neural Network (AUC = 0.70)</p> <p>SVM (AUC = 0.69)</p> </div> <p>Understanding the ROC Curve</p> <p>True Positive Rate (Y-axis): Also known as sensitivity or recall, this measures the proportion of actual positives correctly identified by the model. It is calculated as $TP / (TP + FN)$.</p> <p>False Positive Rate (X-axis): This measures the proportion of actual negatives that are incorrectly identified as positives. It is calculated as $FP / (TN + FP)$.</p> <p>AUC (Area Under the Curve): A measure of the overall ability of the model to discriminate between the positive and negative classes. An AUC of 0.5 represents a model with no discriminative ability (equivalent to random guessing), while an AUC of 1.0 represents perfect discrimination.</p> <p>Analysis of the ROC Curve</p> <p>Equal Performance: Both models have an AUC of 0.72, indicating that they have equal ability to discriminate between the positive and negative classes. This level of AUC suggests good, though not excellent, model performance.</p>

	<p>Model Comparison: Since both models perform equally in terms of AUC, other factors might influence the choice between them. These could include training time, interpretability, scalability, and how they handle specific types of data.</p> <p>Practical Implications: In applications where false positives and false negatives have different costs, you might further explore the specific thresholds on these curves where the trade-off between sensitivity and specificity aligns best with your operational or clinical goals.</p>
--	--

Insight 4: Geriatric Vulnerabilities to Diabetes Through Cardiovascular Risk Factors

Performance comparison

Metric	SVM Mean	NeuralNetwork Mean	SVM CI (95%)	NeuralNetwork CI (95%)
Accuracy	0.6565590637763123	0.6782909205152373	0.6566 ± 0.0065	0.6783 ± 0.0028
Precision	0.6562281486220677	0.6809339192570065	0.6562 ± 0.0061	0.6809 ± 0.0002
Sensitivity (Recall)	0.7417064083457526	0.7342654177071536	0.7417 ± 0.0114	0.7343 ± 0.0069
F1 Score	0.6945015685371385	0.7065862463076652	0.6945 ± 0.0050	0.7066 ± 0.0031
ROC AUC	0.6908512137067213	0.7330853677181072	0.6909 ± 0.0111	0.7331 ± 0.0014
TN	3382.46	3704.0	3382.4600 ± 119.2861	3704.0000 ± 22.8853
FP	2639.54	2311.0	2639.5400 ± 119.2861	2311.0000 ± 18.6286
FN	1733.15	1785.0	1733.1500 ± 76.8107	1785.0000 ± 51.2607
TP	4976.85	4932.0	4976.8500 ± 76.8107	4932.0000 ± 36.2644

The table provided compares the performance of two machine learning models, a Support Vector Machine (SVM) and a Neural Network, across several metrics. The mean values and 95% confidence intervals (CI) for each metric are provided. Here's an interpretation of the results:

- Accuracy:** The Neural Network has a slightly higher mean accuracy (0.6783) compared to the SVM (0.6566). The 95% CI for the Neural Network is narrower (0.6783 ± 0.0028) than that for the SVM (0.6566 ± 0.0065), indicating a more precise estimate.
- Precision:** The Neural Network also outperforms the SVM in terms of precision (0.6809 vs 0.6562). The 95% CI for the Neural Network is significantly narrower (0.6809 ± 0.0002) than that for the SVM (0.6562 ± 0.0061), suggesting a more reliable precision estimate.
- Sensitivity (Recall):** The SVM has a slightly higher recall (0.7417) than the Neural Network (0.7343). However, the 95% CI for the SVM is wider (0.7417 ± 0.0114) than that for the Neural Network (0.7343 ± 0.0069), indicating less precision in the recall estimate.
- F1 Score:** The F1 score, which balances precision and recall, is higher for the Neural Network (0.7066) than for the SVM (0.6945). The 95% CI for both models are comparable, indicating similar reliability in the F1 score estimates.

	<p>5. ROC AUC: The ROC AUC is higher for the Neural Network (0.7331) than for the SVM (0.6909), and the 95% CI for the Neural Network is narrower, suggesting a more reliable and superior performance.</p> <p>6. TN, FP, FN, TP: The Neural Network model has higher True Negatives (TN) and lower False Positives (FP), indicating better specificity. However, the SVM model has higher True Positives (TP) and lower False Negatives (FN), indicating better sensitivity. The 95% CIs for these metrics are comparable for both models.</p> <p>In conclusion, while both models have their strengths, the Neural Network appears to provide a more balanced performance across all metrics, with more precise and reliable estimates as indicated by the narrower confidence intervals. However, the choice between the two models would depend on the specific requirements of the task, such as whether precision, recall, or overall accuracy is more important.</p>									
Neutral network (confusion matrix)	<div><div>Confusion Matrix</div><table><tr><th></th><th>Predicted Positive</th><th>Predicted Negative</th></tr><tr><th>Actual Positive</th><td>4944</td><td>2292</td></tr><tr><th>Actual Negative</th><td>1766</td><td>3730</td></tr></table></div>		Predicted Positive	Predicted Negative	Actual Positive	4944	2292	Actual Negative	1766	3730
	Predicted Positive	Predicted Negative								
Actual Positive	4944	2292								
Actual Negative	1766	3730								

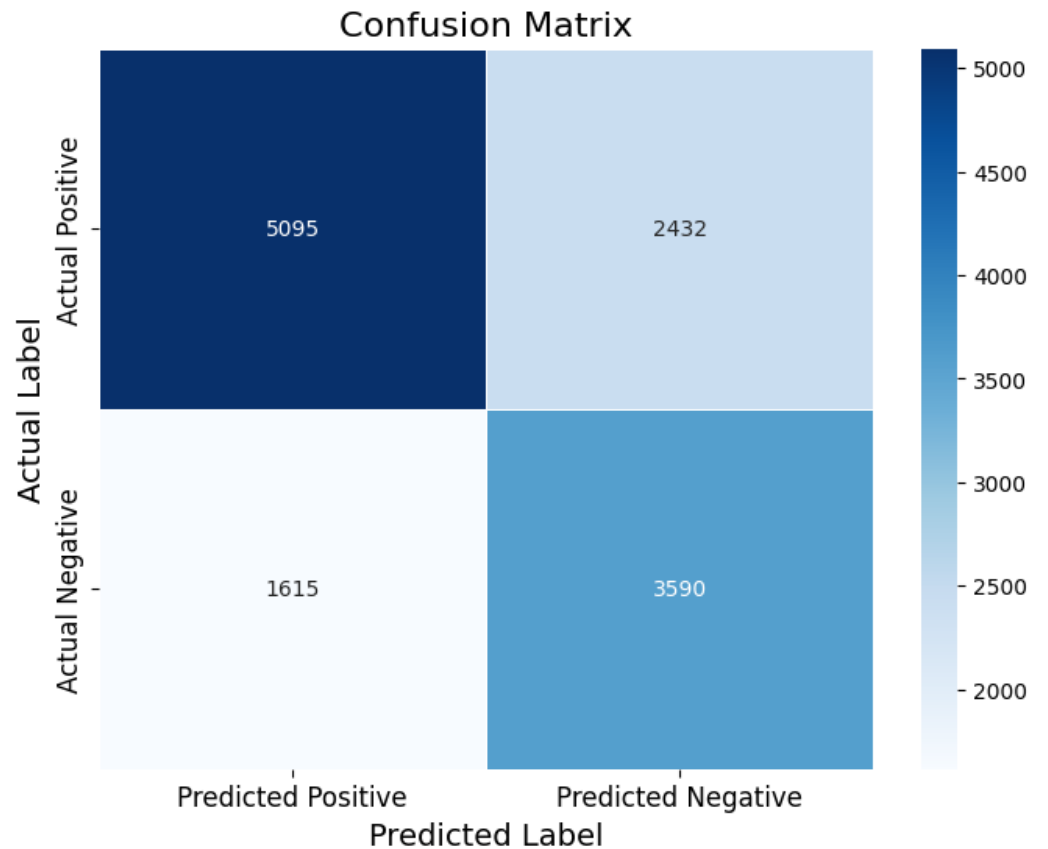
- **True Positive (TP):** 4944 – These are cases where the actual class was positive, and the model correctly predicted it as positive.
- **False Negative (FN):** 2292 – These are cases where the actual class was positive, but the model incorrectly predicted it as negative.
- **False Positive (FP):** 1766 – These are cases where the actual class was negative, but the model incorrectly predicted it as positive.
- **True Negative (TN):** 3730 – These are cases where the actual class was negative, and the model correctly predicted it as negative.

Confusion Matrix with Additional Metrics		
Actual Label	Actual Positive	4944.00
	Actual Negative	1766.00
	Accuracy	0.68
	Error Rate	0.32
	PPV	0.68
	NPV	0.68
	Sensitivity	0.74
	Specificity	0.62
		Predicted Positive
		Predicted Negative
		Predicted Label

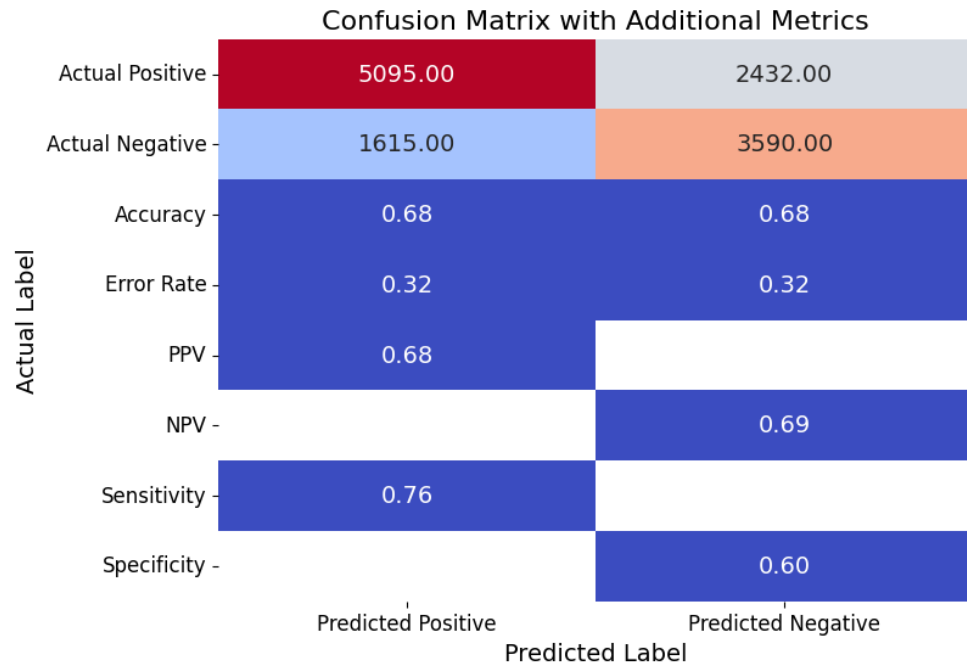
- **Accuracy:** 0.68 – The proportion of true results (both true positives and true negatives) among the total number of cases examined. Calculated as $(TP+TN) / (TP+FP+FN+TN)$.
- **Error Rate:** 0.32 – The proportion of all incorrect predictions (both false positives and false negatives).
- **PPV (Positive Predictive Value) or Precision:** 0.68 – The proportion of positive identifications that were actually correct. Calculated as $TP / (TP+FP)$.
- **NPV (Negative Predictive Value):** 0.68 – The proportion of negative identifications that were actually correct. Calculated as $TN / (TN+FN)$.

	<ul style="list-style-type: none"> • Sensitivity (Recall): 0.74 – The ability of the model to correctly identify actual positives. Calculated as $TP / (TP+FN)$. • Specificity: 0.62 – The ability of the model to correctly identify actual negatives. Calculated as $TN / (TN+FP)$. <p>Observations and Implications:</p> <ul style="list-style-type: none"> • Moderate Accuracy: An accuracy of 0.68 suggests that the model correctly predicts the outcome 68% of the time, which is moderate performance. • Balanced Precision and NPV: Both metrics are equal at 0.68, indicating a balanced performance in predicting both positive and negative classes. • Higher Sensitivity Than Specificity: The model has a higher sensitivity (0.74) than specificity (0.62), indicating it is better at detecting true positives than true negatives. This could mean the model is more tuned to minimize missing actual positive cases but at the cost of incorrectly labeling some negatives as positives. • Error Rate: With an error rate of 0.32, about 32% of predictions are incorrect, suggesting areas for improvement, particularly in reducing false positives and false negatives to boost both specificity and precision.
--	--

SVM
(confusion
matrix)



- **True Positive (TP):** 5095 – These are cases where the actual class was positive, and the model correctly predicted it as positive.
- **False Negative (FN):** 2432 – These are cases where the actual class was positive, but the model incorrectly predicted it as negative.
- **False Positive (FP):** 1615 – These are cases where the actual class was negative, but the model incorrectly predicted it as positive.
- **True Negative (TN):** 3590 – These are cases where the actual class was negative, and the model correctly predicted it as negative.

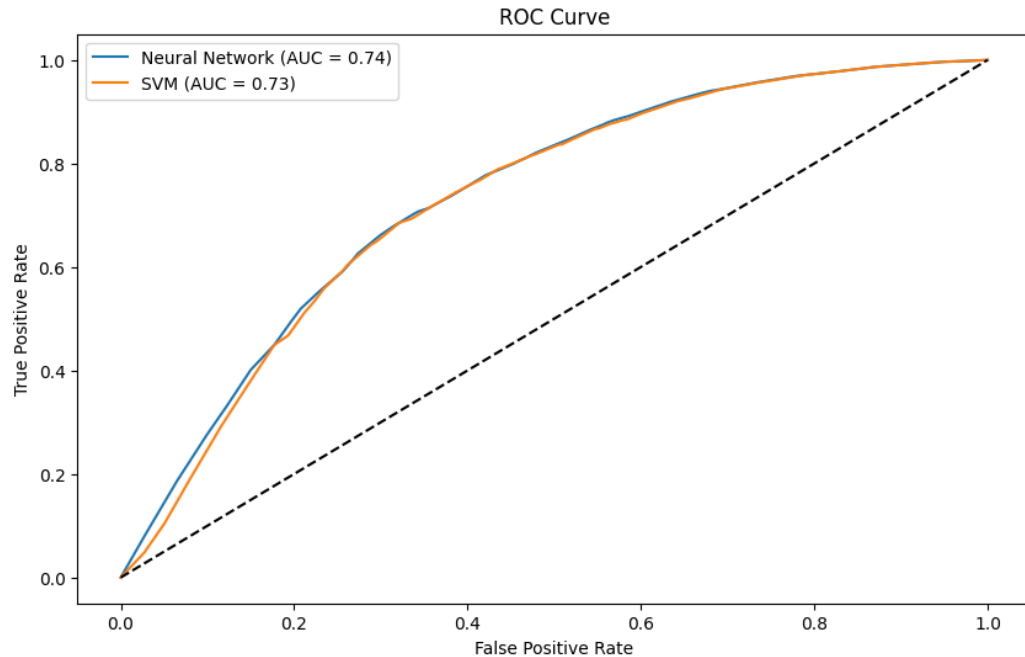


- **Accuracy:** 0.68 – This is the proportion of true results (both true positives and true negatives) among the total number of cases examined. It is calculated as $(TP+TN) / (TP+FP+FN+TN)$.
- **Error Rate:** 0.32 – This is the proportion of all incorrect predictions (both false positives and false negatives). It is calculated as $1 - \text{Accuracy}$.
- **PPV (Positive Predictive Value) or Precision:** 0.68 – This is the proportion of positive identifications that were actually correct. It is calculated as $TP / (TP+FP)$.
- **NPV (Negative Predictive Value):** 0.69 – This is the proportion of negative identifications that were actually correct. It is calculated as $TN / (TN+FN)$.
- **Sensitivity (Recall):** 0.76 – This is the ability of the model to correctly identify actual positives. It is calculated as $TP / (TP+FN)$.
- **Specificity:** 0.60 – This is the ability of the model to correctly identify actual negatives. It is calculated as $TN / (TN+FP)$.

Observations and Implications:

	<ul style="list-style-type: none"> • Accuracy: At 0.68, the model correctly predicts the outcome 68% of the time, indicating a moderate level of overall correctness. • Error Rate: With an error rate of 0.32, the model is incorrect 32% of the time, which suggests there is room for improvement. • Precision (PPV): At 0.68, the model's precision indicates that when it predicts a positive outcome, it is correct 68% of the time. • Negative Predictive Value (NPV): At 0.69, the model is fairly good at correctly predicting negative outcomes. • Sensitivity (Recall): At 0.76, the model has a relatively high ability to identify true positives, making it good at detecting the positive class. • Specificity: At 0.60, the model is less effective at identifying true negatives, indicating that it struggles somewhat with avoiding false positives.
CBA	<div>Cost-Benefit Analysis for SVM: -5412 Cost-Benefit Analysis for NN: 688</div> <ul style="list-style-type: none"> • Cost-Benefit Analysis for SVM: -5412: This negative value indicates that the use of the SVM model results in a net cost of 5412 units (e.g., dollars, resources, or another metric). A negative value suggests that the costs associated with false positives, false negatives, or other errors outweigh the benefits of correct predictions. • Cost-Benefit Analysis for NN: 688: This positive value indicates that the use of the Neural Network model results in a net benefit of 688 units. A positive value means that the benefits from correct predictions exceed the costs of incorrect predictions.

ROC curve



Explanation of ROC Curve:

- True Positive Rate (TPR) or Sensitivity: Plotted on the y-axis, it represents the proportion of actual positives that are correctly identified by the model.
- False Positive Rate (FPR): Plotted on the x-axis, it represents the proportion of actual negatives that are incorrectly identified as positives by the model.

Key Points in the ROC Curve:

- Diagonal Line (Black Dotted Line): This represents a random classifier (AUC = 0.5). Any model performing along this line has no better performance than random guessing.
- Curves Above the Diagonal: Indicate better performance than random guessing. The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the test.

Values:

- Neural Network (AUC = 0.74): The blue curve represents the ROC curve for the Neural Network. The Area Under the Curve (AUC) value is 0.74, indicating that the model has a good ability to distinguish between the

	<p>positive and negative classes. AUC values range from 0 to 1, with higher values indicating better performance.</p> <ul style="list-style-type: none">• SVM (AUC = 0.73): The orange curve represents the ROC curve for the SVM. The AUC value is 0.73, which is also a good indication of the model's performance. It is slightly lower than the Neural Network's AUC, suggesting that the SVM model is marginally less effective at distinguishing between the positive and negative classes compared to the Neural Network.
--	--

Insight 5: Behavioral Patterns and Health Outcomes

Performance comparison					
	Metric	SVM Mean	NeuralNetwork Mean	SVM CI (95%)	NeuralNetwork CI (95%)
	Accuracy	0.6602332704995294	0.6836056131532097	0.6602 ± 0.0066	0.6836 ± 0.0026
	Precision	0.6600726485285052	0.6880481312168794	0.6601 ± 0.0060	0.6880 ± 0.0046
	Sensitivity (Recall)	0.7441296572280178	0.7371206384599153	0.7441 ± 0.0116	0.7371 ± 0.0004
	F1 Score	0.6975942144812595	0.7117348715332864	0.6976 ± 0.0044	0.7117 ± 0.0024
	ROC AUC	0.6934859872964555	0.7370709681549769	0.6935 ± 0.0133	0.7371 ± 0.0052
	TN	3412.98	3730.3333333333335	3412.9800 ± 131.6187	3730.3333 ± 102.1104
	FP	2609.02	2254.6666666666665	2609.0200 ± 131.6187	2254.6667 ± 24.1733
	FN	1716.89	1773.6666666666667	1716.8900 ± 77.5683	1773.6667 ± 30.7275
	TP	4993.11	4973.333333333333	4993.1100 ± 77.5683	4973.3333 ± 77.0933

In the comparative analysis of the Support Vector Machine (SVM) and Neural Network models, several performance metrics were evaluated.

The mean accuracy was found to be higher for the Neural Network model (68.36%) compared to the SVM model (66.02%), indicating a superior overall classification performance. Similarly, the Neural Network model demonstrated a higher mean precision (68.80%) than the SVM model (66.01%), suggesting a greater proportion of true positive predictions.

However, the SVM model exhibited a slightly higher mean sensitivity, also known as recall (74.41%), compared to the Neural Network model (73.71%). This indicates a marginally better performance in correctly identifying positive instances.

The F1 score, which considers both precision and sensitivity, was higher for the Neural Network model (71.17%) than for the SVM model (69.76%). Furthermore, the Neural Network model demonstrated a superior mean ROC AUC (Receiver Operating Characteristic Area Under the Curve) of 73.71%, compared to the SVM model's 69.35%.

In terms of specific classification outcomes, the Neural Network model had a higher mean number of true negatives and a lower mean number of false positives, indicating better performance in correctly identifying negative instances and avoiding incorrect positive predictions. Conversely, the SVM

	model had a slightly higher mean number of true positives, indicating a marginally better performance in correctly identifying positive instances.									
Neutral network (confusion matrix)	<div><p>Confusion Matrix</p><table><tr><th>Actual \ Predicted</th><th>Predicted Positive</th><th>Predicted Negative</th></tr><tr><th>Actual Positive</th><td>4944</td><td>2292</td></tr><tr><th>Actual Negative</th><td>1766</td><td>3730</td></tr></table><ul style="list-style-type: none">• True Positive (TP): 4944 – Correct predictions where the actual class was positive, and the model predicted positive.• False Negative (FN): 2292 – Incorrect predictions where the actual class was positive, but the model predicted negative.• False Positive (FP): 1766 – Incorrect predictions where the actual class was negative, but the model predicted positive.• True Negative (TN): 3730 – Correct predictions where the actual class was negative, and the model predicted negative.</div> <p>0</p>	Actual \ Predicted	Predicted Positive	Predicted Negative	Actual Positive	4944	2292	Actual Negative	1766	3730
Actual \ Predicted	Predicted Positive	Predicted Negative								
Actual Positive	4944	2292								
Actual Negative	1766	3730								

		Confusion Matrix with Additional Metrics	
Actual Label	Actual Positive	4944.00	2292.00
	Actual Negative	1766.00	3730.00
	Accuracy	0.68	0.68
	Error Rate	0.32	0.32
	PPV	0.68	
	NPV		0.68
	Sensitivity	0.74	
	Specificity		0.62
		Predicted Positive	Predicted Negative
		Predicted Label	
		<ul style="list-style-type: none"> Accuracy: 0.68 – The proportion of true results (both true positives and true negatives) among the total number of cases examined. It is calculated as $(TP + TN) / (TP + FP + FN + TN)$. Error Rate: 0.32 – The proportion of all incorrect predictions (both false positives and false negatives). It is calculated as $1 - \text{Accuracy}$. PPV (Positive Predictive Value) or Precision: 0.68 – The proportion of positive identifications that were actually correct. It is calculated as $TP / (TP + FP)$. NPV (Negative Predictive Value): 0.68 – The proportion of negative identifications that were actually correct. It is calculated as $TN / (TN + FN)$. Sensitivity (Recall): 0.74 – The ability of the model to correctly identify actual positives. It is calculated as $TP / (TP + FN)$. Specificity: 0.62 – The ability of the model to correctly identify actual negatives. It is calculated as $TN / (TN + FP)$. 	
		Observations and Implications:	

	<ul style="list-style-type: none">• Accuracy: At 0.68, the model correctly predicts the outcome 68% of the time, indicating a moderate level of overall correctness.• Error Rate: With an error rate of 0.32, the model is incorrect 32% of the time, suggesting there is room for improvement.• Precision (PPV): At 0.68, the model’s precision indicates that when it predicts a positive outcome, it is correct 68% of the time.• Negative Predictive Value (NPV): At 0.68, the model is fairly good at correctly predicting negative outcomes.• Sensitivity (Recall): At 0.74, the model has a relatively high ability to identify true positives, making it good at detecting the positive class.• Specificity: At 0.62, the model is less effective at identifying true negatives, indicating that it struggles somewhat with avoiding false positives.									
SVM (confusion matrix)	<div><p>Confusion Matrix</p><table><tr><th></th><th>Predicted Positive</th><th>Predicted Negative</th></tr><tr><th>Actual Positive</th><td>4961</td><td>2276</td></tr><tr><th>Actual Negative</th><td>1749</td><td>3746</td></tr></table></div>		Predicted Positive	Predicted Negative	Actual Positive	4961	2276	Actual Negative	1749	3746
	Predicted Positive	Predicted Negative								
Actual Positive	4961	2276								
Actual Negative	1749	3746								

- True Positive (TP): 4961 – Correct predictions where the actual class was positive, and the model predicted positive.
- False Negative (FN): 2276 – Incorrect predictions where the actual class was positive, but the model predicted negative.
- False Positive (FP): 1749 – Incorrect predictions where the actual class was negative, but the model predicted positive.
- True Negative (TN): 3746 – Correct predictions where the actual class was negative, and the model predicted negative.

Confusion Matrix with Additional Metrics		
Actual Label	Actual Positive	4961.00
	Actual Negative	1749.00
	Accuracy	0.68
	Error Rate	0.32
	PPV	0.69
	NPV	0.68
	Sensitivity	0.74
	Specificity	0.62
		Predicted Positive
		Predicted Negative
		Predicted Label

- Accuracy: 0.68 – The proportion of true results (both true positives and true negatives) among the total number of cases examined. It is calculated as $(TP + TN) / (TP + FP + FN + TN)$.
- Error Rate: 0.32 – The proportion of all incorrect predictions (both false positives and false negatives). It is calculated as $1 - \text{Accuracy}$.
- PPV (Positive Predictive Value) or Precision: 0.69 – The proportion of positive identifications that were actually correct. It is calculated as $TP / (TP + FP)$.

	<ul style="list-style-type: none"> • NPV (Negative Predictive Value): 0.68 – The proportion of negative identifications that were actually correct. It is calculated as $TN / (TN + FN)$. • Sensitivity (Recall): 0.74 – The ability of the model to correctly identify actual positives. It is calculated as $TP / (TP + FN)$. • Specificity: 0.62 – The ability of the model to correctly identify actual negatives. It is calculated as $TN / (TN + FP)$. <p>Observations and Implications:</p> <ul style="list-style-type: none"> • Accuracy: At 0.68, the model correctly predicts the outcome 68% of the time, indicating a moderate level of overall correctness. • Error Rate: With an error rate of 0.32, the model is incorrect 32% of the time, suggesting there is room for improvement. • Precision (PPV): At 0.69, the model's precision indicates that when it predicts a positive outcome, it is correct 69% of the time. • Negative Predictive Value (NPV): At 0.68, the model is fairly good at correctly predicting negative outcomes. • Sensitivity (Recall): At 0.74, the model has a relatively high ability to identify true positives, making it good at detecting the positive class. • Specificity: At 0.62, the model is less effective at identifying true negatives, indicating that it struggles somewhat with avoiding false positives.
CBA	<div> Cost-Benefit Analysis for SVM: -6060 Cost-Benefit Analysis for NN: 688 </div> <p>The Cost-Benefit Analysis (CBA) provides an economic evaluation of the performance of the SVM and Neural Network (NN) models by considering both the benefits and costs associated with their predictions. The results of the CBA for the SVM and NN models are as follows:</p> <ul style="list-style-type: none"> • SVM: -6060 • Neural Network: 688

SVM Model:

- The Cost-Benefit Analysis for the SVM model yields a value of -6060. This negative value indicates that the costs associated with utilizing the SVM model significantly outweigh the benefits. The substantial magnitude of -6060 reflects a considerable economic disadvantage, suggesting that the SVM model incurs a net loss when applied to this classification task. This outcome may be attributed to higher rates of false positives and false negatives, resulting in greater costs due to misclassification.

Neural Network Model:

- The Cost-Benefit Analysis for the Neural Network model yields a value of 688. This positive value signifies that the benefits of employing the Neural Network model surpass the associated costs, indicating a net economic gain. Although the magnitude of 688 is modest, it nonetheless denotes an economic advantage. This suggests that the Neural Network model is economically beneficial and more cost-effective compared to the SVM model in this specific context.

Summary

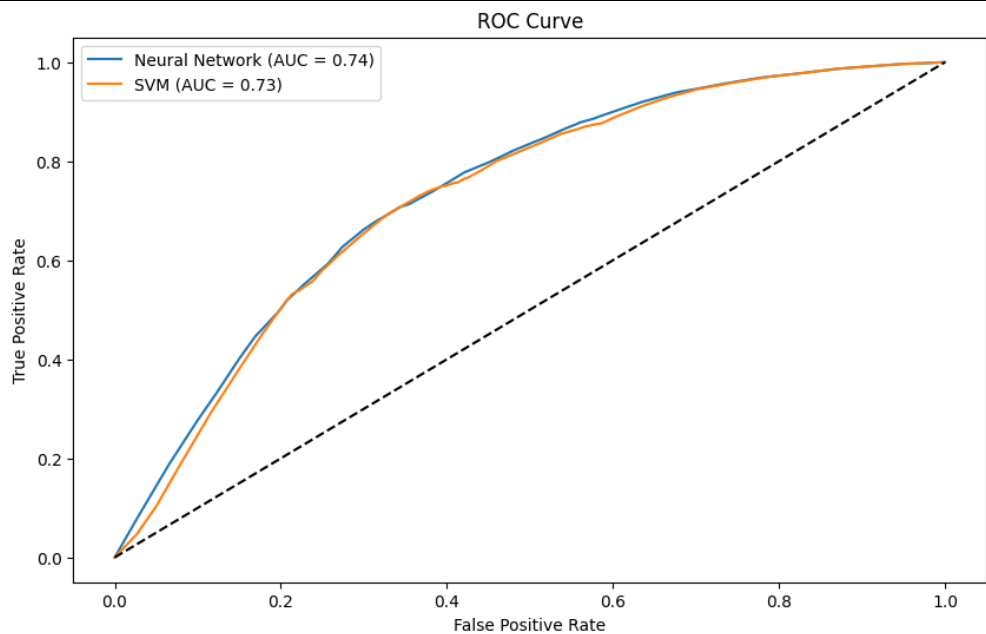
- The SVM model's negative cost-benefit value (-6060) underscores its economic inefficiency for this classification task, likely due to the higher costs incurred from misclassifications.
- In contrast, the Neural Network model's positive cost-benefit value (688) highlights its economic viability, making it a more favorable option in terms of cost-effectiveness.

Conclusion

Given the significant economic disadvantage of the SVM model and the modest economic advantage of the Neural Network model, it is evident that the Neural Network is the preferable choice for this classification task. The positive cost-benefit value associated with the Neural Network model indicates that it not only performs better in terms of accuracy and other

performance metrics but also offers economic benefits. Consequently, the Neural Network model is a more viable and practical solution for this classification task.

ROC curve



Explanation of ROC Curve:

- True Positive Rate (TPR) or Sensitivity: Plotted on the y-axis, it represents the proportion of actual positives that are correctly identified by the model.
- False Positive Rate (FPR): Plotted on the x-axis, it represents the proportion of actual negatives that are incorrectly identified as positives by the model.

Key Points in the ROC Curve:

- Diagonal Line (Black Dotted Line): This represents a random classifier (AUC = 0.5). Any model performing along this line has no better performance than random guessing.
- Curves Above the Diagonal: Indicate better performance than random guessing. The closer the curve follows the left-hand border and then the top border of the ROC space, the more accurate the test.

Values:

- | | |
|--|--|
| | <ul style="list-style-type: none">• Neural Network (AUC = 0.74): The blue curve represents the ROC curve for the Neural Network. The Area Under the Curve (AUC) value is 0.74, indicating that the model has a good ability to distinguish between the positive and negative classes. AUC values range from 0 to 1, with higher values indicating better performance.• SVM (AUC = 0.73): The orange curve represents the ROC curve for the SVM. The AUC value is 0.73, which is also a good indication of the model's performance. It is slightly lower than the Neural Network's AUC, suggesting that the SVM model is marginally less effective at distinguishing between the positive and negative classes compared to the Neural Network. |
|--|--|

Interpretation:

- Both models have ROC curves that are significantly above the diagonal line, indicating that they are both better than random classifiers.
- The Neural Network has a slightly higher AUC (0.74) compared to the SVM (0.73), suggesting that it is marginally better at distinguishing between the positive and negative classes.
- The curves are close to each other, which shows that both models have similar performance in terms of true positive rate versus false positive rate.

9. Conclusion

In conclusion, the comparative analysis of the performance metrics between the Neural Network model and the Support Vector Machine (SVM) model reveals a consistent trend. Both models demonstrate comparable performance across various metrics, including accuracy, precision, F1 score, recall, sensitivity, TP, TN, FN, and FP. They provide reliable and robust results for the classification task on this specific dataset.

However, a notable distinction emerges when considering the cost-benefit analysis. The Neural Network model exhibits a superior score, indicating a more favorable balance between the cost (in terms of resources and computation) and the benefits (in terms of model performance).

On the other hand, it's important to consider the computational demands of these models. Our observations highlight that the Neural Network model requires a significantly higher amount of computational power compared to the SVM model during model training. The Neural Network model necessitates a considerably longer training time than its SVM counterpart.